Topic Classification for Short Texts

Dan Claudiu Neagu
Cicada Technologies and Babes-Bolyai University
Cluj-Napoca, Romania *dann@cicadatech.eu*

Andrei Bogdan Rus
Cicada Technologies and Technical University of Cluj-Napoca
Cluj-Napoca, Romania *bogdanr@cicadatech.eu*

Mihai Grec
Cicada Technologies
Cluj-Napoca, Romania *mihaig@cicadatech.eu*

Mihai Boroianu
Cicada Technologies
Cluj-Napoca, Romania *mihaib@cicadatech.eu*

Gheorghe Cosmin Silaghi
Babes-Bolyai University
Cluj-Napoca, Romania *gheorghe.silaghi@ubbcluj.ro*

Abstract

In the context of TV and social media surveillance, constructing models to automate topic identification of short texts is a key task. This paper constructs worth-to-consider models for practical usage, employing Top-K multinomial classification methodology. We describe the full data processing pipeline, discussing about dataset selection, text preprocessing, feature extraction, model selection and learning, including hyperparameter optimization. We will test and compare popular methods including: standard machine learning, deep learning, and a fine-tuned BERT for topic classification.

Keywords: Natural Language Processing, Topic Classification, Text Classification, Machine Learning, Deep Learning

1. Introduction

Part of Natural Language Processing (NLP), document classification assigns a document to one or more classes or categories. When the input is text, we speak about text classification, which is a popular technique applied to automate various processes like spam filtering [21], humor detection [16], sentiment analysis [27] and many more including topic classification. This last term refers to identifying abstract topics that occur in a collection of texts or documents, with the motivation of discovering the interests discusses or described within the textual data.

Opposed to sentiment analysis where the overall target is to determine the polarity of a text (i.e. positive, neutral or negative), in topic classification the number of classes could be extremely high and in many cases, overlapping, thus the problem becoming much more difficult [18, 27]. Rather than considering topic classification as a standard classification problem, we employ a *Top-K* multinomial classification methodology [20, 25], with the goal of inferring the k highest probable classes for each text, from a set of predefined topics.

Special focus is towards social media, as its texts are of great informative value. In general, social media platforms disseminate short unstructured texts, generated by their authors from mobile devices in short time intervals, with plenty of bad language [15]. These characteristics of social media inputs bring in additional challenges, because such a short text does not provide sufficient word occurrences for a traditional text classification based on "Bag-Of-Words" document representations [22].

The work presented here is done under the umbrella of a media surveillance project [13], aiming to investigate specific habits of people interacting with TV and social media. Several restrictions and limitations are imposed on the project like: frequent model retraining and deployment due to the volatile nature of the environment, the need to processed immense volumes of data in short time intervals, and the need to adhere to data privacy laws and security standards. Given that, this paper describes our efforts for building a worth-to-consider classifier for short texts which can be applied on social media data. Besides searching for a good topic prediction accuracy of the various classifiers, the processing time is also highly important.

We present in detail the steps involved in the full NLP pipeline, from raw texts to topic prediction, and the Top-1, Top-2, and Top-3 accuracies achieved by various classifiers. The pipeline contains the following processes: dataset selection, data cleaning and preprocessing, feature extraction, training and testing various classical machine learning and deep learning models, and the hyper-parameter optimization methodology used for identifying the best parameters for each trained ML model.

The rest of the paper is structured as following: in section 2 we present related work competing or influencing our research. Section 3 introduces the data under study and the steps followed to construct the topic classification models. Section 4 present the achieved results and section 5 concludes our work.

2. Related Work

Discovering abstract topics that occur in a collection of texts or documents could be done with either *Topic Classification* or *Topic Modeling*. Topic modeling is an unsupervised technique [8, 45] that doesn't require labeled data, while topic classification is a supervised one, where labeled data is needed for model training.

Topic modeling is a popular statistical tool for extracting latent variables from large datasets, being well suited for textual data [8, 45]. Among the most used methods for topic modeling we mention Probabilistic Latent Semantic Analysis (PSLA) and Latent Dirichlet Allocation (LDA). In essence, conventional topic models reveal topics within a text corpus by implicitly capturing the document-level word co-occurrence patterns [47, 9]. Directly applying these models on short texts will suffer from the severe data sparsity problem, i.e. the sparse word co-occurrence patterns in individual document [22]. Some workarounds try to alleviate the sparsity problem. Albanese & Feuerstein aggregate a number of short texts to create a lengthy pseudo-document [2], its effectiveness being heavily data-dependent. The Biterm Topic Model [11] extracts unordered word pairs (i.e. biterms) co-occuring in short texts and the latent topic components being modeled using these biterms. This method seems to perform better for short texts compared to other traditional approaches.

The main advantage of topic modeling methods is that they do not require labeled data, thus data collection becomes more accessible and could be done in a fully or partially automated manner. Despite its popularity, topic modeling is prone to serious issues with optimization, noise sensitivity or result instability [1] and some techniques are not being representative for real-world data relationships [7].

If labeled training data is available, *topic classification* overcomes most issues related with topic modeling. Learning models on a small dataset with around 770 tweets distributed over 18 classes, Lee et al. [26] achieved an accuracy of $\approx 65\%$ with the multinomial Naive Bayes classifier and $\approx 62\%$ with the standard SVM classifier. Rahman & Akter [41] worked with

6000 texts extracted from *Amazon's product review corpus*[1] distributed over only 6 very specific topics and achieved a very high classification rate of $\approx 92\%$ with NB, $\approx 82\%$ with k-NN and $\approx 79\%$ with decision trees. Zeng et al. [48] proposes a hybrid approach, extracting the most relevant latent features with topic modeling and then, feeding them to supervised ML algorithms like SVM, CNN and LSTM. For the experiments they used the Twitter dataset released by TREC2011[2] with around 15000 tweets, semi-automatically labeled into 50 topic classes. The best obtained accuracy is with CNN and is poor: only $\approx 39.5\%$, and the topic modeling seems not to significantly improve the learning.

Difficulty of topic classification resides also in the big number of target classes. To overcome this, some authors [20, 34] use the Top-K accuracy instead of the standard one. Rather than classifying a text to just one class and matching it to the a-priori label, the model will produce the most K probable classes and if the label is among them, we consider the text as being correctly classified. In our work we will report the standard accuracy (i.e. Top-1), the Top-2 and Top-3 accuracies.

3. Data Processing Methodology

In the following subsections we present the whole pipeline methodology for topic classification. We start by describing the dataset and after that we present the full processing pipeline: data preprocessing, feature extraction, classifier selection and model building with the help of hyperparameter optimization.

All processing modules were implemented in Python 3.9 and experiments were run on a powerful machine with the following specifications: 2 $\times$ Intel Xeon Gold 6230 CPUs (20 Core at 2.1 GHz), 128 GB DDR4 internal RAM, 8 $\times$ NVIDIA Tesla V100 32 GB.

3.1. Dataset Description

The dataset selected for our topic classification experiments is the *News Category dataset* available on Kaggle[3] [31]. This dataset contains 202372 news headlines collected between 2012 up to 2018 from HuffPost[4] The site offers news, satire, blogs, original content, and covers a variety of topics like politics, business, entertainment, technology, popular media, and more.

There are a number of reasons why we selected this dataset as the benchmark for our experiments: (i) it contains short texts similar to those found on social media platforms, (ii) the topics are fairly general and the number of topics is large enough, (iii) the category of each article was manually labeled, (iv) high data volume, and (v) it was relatively recently collected. Each record of the dataset contains the following attributes: *category* (41 categories), *headline*, *short_description*, *authors*, *date* (of the publication), and *link* (URL link of the article).

For our classification problem we will focus only on the headline and short description attributes of the dataset, ignoring the authors and date of publication. Therefore, we merged the headline and the short description attributes and created a novel attribute named *text_merged*.

The vast majority of merged texts contain between 94 and 254 characters, with the mean being ≈ 174 and the standard deviation almost 80 characters. This proves that the generated texts have the characteristics of short texts similar to those present in social media platforms (a Twitter tweet is limited to 280 characters, a Youtube comment is limited to 300 characters).

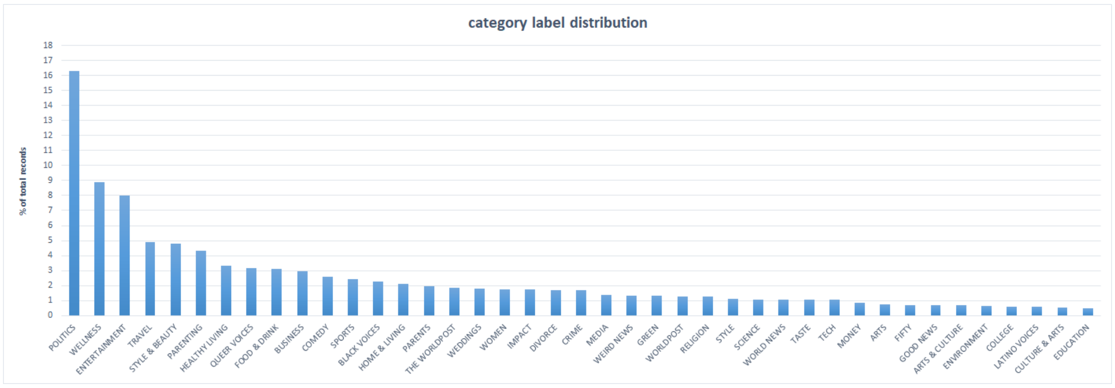
Figure 1a shows the distribution of the records among the 41 categories. The top-3 most popular classes are: "POLITICS" which contains $\approx 16\%$ of the records, "WELLNESS" which con-

[1]`https://jmcauley.ucsd.edu/data/amazon/`

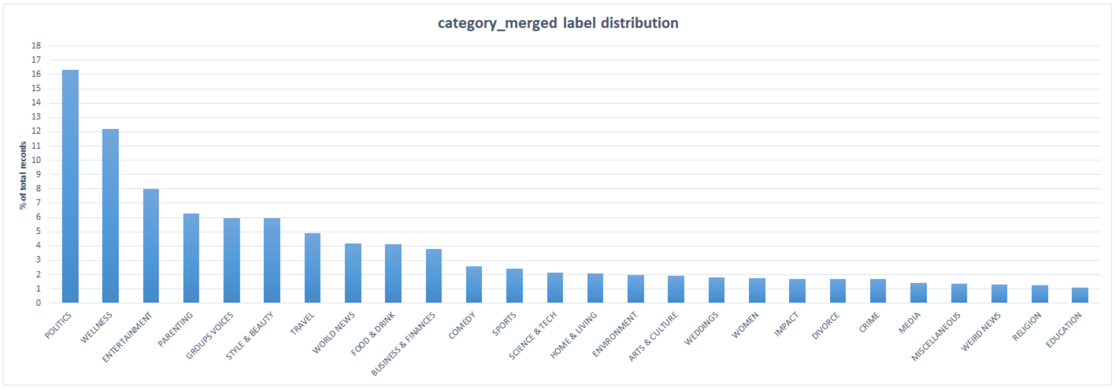
[2]`http://trec.nist.gov/data/tweets`

[3]`https://www.kaggle.com/datasets/rmisra/news-category-dataset`

[4]`https://www.huffpost.com/`, formerly The Huffington Post until 2017, is an American news aggregator and blog with localized and international editions.



(a) Original category distribution (41 total classes)



(b) Merged category distribution (26 total classes)

Fig. 1. Class distributions for the News Category dataset.

tains $\approx 9\%$ of the records, and "ENTERTAINMENT" which contains $\approx 8\%$ of the records. The least most popular 4 classes are: "COLLEGE", "LATINO VOICES", "CULTURE & ARTS", and "EDUCATION" each containing around 0.5% of the records.

Some classes have different labels but denote the same topic, for example the classes "ARTS & CULTURE" and "CULTURE & ARTS". Other classes are highly granular, like "SCIENCE" and "TECH" but can be naturally grouped together in a common class. In order to improve the quality of the data, we decided to cluster together a number of classes. Therefore, we transformed the following classes as follows: "HEALTHY LIVING" was relabeled as the existing "WELLNESS" class; "PARENTS" was relabeled as the existing "PARENTING" class; "STYLE" was relabeled as the existing "STYLE & BEAUTY" class; "GREEN" was relabeled as the existing "ENVIRONMENT" class; "TASTE" was relabeled as the existing "FOOD & DRINK" class; "COLLEGE" was relabeled as the existing "EDUCATION" class; "THE WORLDPOST" and "WORDPOST" were relabeled as the existing "WORLD NEWS" class; "ARTS" and "CULTURE & ARTS" were relabeled as the existing "ARTS & CULTURE" class; "BUSINESS" and "MONEY" were relabeled as a new class named "BUSINESS & FINANCES"; "SCIENCE" and "TECH" were relabeled as a new class named "SCIENCE & TECH"; "QUEER VOICES", "BLACK VOICES", and "LATINO VOICES" were relabeled as a new class named "GROUPS VOICES"; "FIFTY" and "GOOD NEWS" were relabeled as a new class named "MISCELLANEOUS".

The new class feature was named *category_merged* and it contains only 26 distinct topics, compared to the original 41. The full distribution of the merged topics can be seen in Figure 1b. After the topic clustering, no class has less than 1% of record labels, meaning that the least popular class has more than 2000 records in the dataset.

3.2. Text Preprocessing

In order to remove the natural noise which is existent in textual data, we developed a specialized module for preprocessing (PP). It contains various functionalities which can be applied on many types of textual data, not only on short texts. Some of them relies on SpaCy[5] library [23]. For the News Category dataset we applied the following preprocessing steps, in this specific order:

1. Extra white space removal
2. Word lemmatisation and tokenization
3. Stop-word removal
4. Lower case capitalization
5. Punctuation mark removal

In step 1, all consecutive white spaces which appear more than two times are removed from the texts, i.e. "Foo   bar!" becomes "Foo bar!". In step 2, we performed word lemmatisation and tokenization, as advised by [33]. The input for this step are strings of the *text_merged* attribute and the generated output is the list of tokens, where each token is either a number, lemmatised word, or symbol. In step 3, stop-word tokens are identified and removed using the stop-word dictionary offered by SpaCy. Stop-word removal is a common task in text preprocessing, as indicated by [27]. In step 4, all tokens are transformed in lower case capitalization in order to reduce the number of tokens which are written in different capitalizations but refer to the same concept. The final step 5 removes all extra punctuation marks within tokens.

For the BERT classifier, a single preprocessing step consisting of sentence tokenization was applied.

After the preprocessing step, the dataset was split into training and testing sets. The training set contains 75% of the data while the testing test contains the remaining 25%. The split was made such that the class distribution between the train and test set is similar.

3.3. Feature Extraction

Feature extraction is a key process of every NLP task. It starts from an initial set of text data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps [18, 29]. Here, we considered *Term Frequency-Inverse Document Frequency (TFIDF or TF-IDF)* [43] and *Word2Vec* [30]. TFIDF is still widely used today [4] together with classical ML methods, although it can not account for the similarity between the words in the document and in general, Word2Vec is seen as facilitating deep learning.

We trained a TFIDF vectorizer on the training set. The TFIDF was applied on the preprocessed token lists and the vocabulary was set to contain the tokens which appear at least 5 times. This was done in order to remove a large number of tokens which are very rarely used or tokens which may have been erroneously built in the preprocessing step. The trained TFIDF vocabulary contains around 25000 tokens. Due to the sparse nature of TFIDF, the large number of training instances, and the vocabulary size, the trained vectors are stored and used in the *Compressed Sparse Row (CRS)* format.

Word2Vec[30] uses a neural network model to learn word associations from a large corpus of texts, being able to better capture the language semantics. We used the Gensim[6] library [42] for learning the word vectors from our training data and we finally kept only the tokens which

[5]https://spacy.io/

[6]https://radimrehurek.com/gensim/

appear at least 5 times. The Word2Vec algorithm was applied using the Continuous Bag-Of-Words (CBOW) architecture model. The vector embedding size for each token was set to 300. The model was trained with the following parameters: learning rate *alpha* of 0.025, *window* of 5, over 5 *epochs*. After this step, each token from the vocabulary is represented by a feature vector of size 300.

In order to fine-tune the BERT classifier, the pre-trained BertTokenizer[7] was applied on the tokenized sentences. This process generated the specific BERT encodings consisting of word mappings and an attention masks.

3.4. Classifier Selection

Many supervised classifiers have been applied for text classification tasks [18, 27]. For our experiments we selected the following methods for classification:

- Classic ML: Bernoulli Naive Bayes (Bernoulli NB), Random Forest, Support Vector Machine (SVM)
- Deep Learning: Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN)
- BERT plus a classification layer

Bernoulli NB[28] is a probabilistic classifier designed to use an implicit mixture model for generation of the underlying documents. It can handle irrelevant or missing features which are very common in text classification and in some NLP tasks this is the reference model [29]. Due to its simplistic nature, model training times are usually very low. Random forest [10] is a popular choice for classification tasks over a large features set, including text classification[18]. SVMs [25] are reported to achieve state of the art prediction performances for text classification compared to other classic ML algorithms. Thus, we selected SVM among our classifiers and we trained the models with a linear kernel in order to better scale to large number of samples and to reduce the training time.

Deep Learning (DL) with the help of artificial neural networks has achieved state of the art results across many domains, including a wide variety of NLP applications. Thus, we selected two variations of DL for our task: the Long short-term memory (LSTM) network [36] as a type of recurrent neural networks (RNNs) and the Convolutional Neural Network (CNN) [24].

The BERT model available on the Hugging Face transformers[8], was used with the standard English uncased variant. On top of BERT we added a simple neural network with one hidden dense layer with 128 nodes and ReLU activation function, followed by the standard classification layer with 26 nodes which will generate the topic. We selected Adam as the optimization function, with a learning rate of $2e-5$ and $\epsilon = 1e-8$. The loss function was set to Categorical Cross-Entropy. The BERT classifier was fine-tuned on the specific BERT encodings generated in section 3.3.

Compared to the standard ML algorithms presented above, LSTMs and CNNs are able to process sequences of data as is the case with the multidimensional representation of Word2Vec. Due to this, the LSTM and CNN algorithms were applied on the Word2Vec features while the classic learning algorithms (Bernoulli NB, Random Forest, and SVM) were applied only on the TFIDF features. In order to apply LSTM and CNN on the sequenced data, we introduced an embedding layer between the input layer and the hidden layers. The embedding layer maps each token from an instance to its corresponding Word2Vec representation, being equipped with the word embeddings generated during feature extraction (see section 3.3).

[7]`https://huggingface.co/docs/transformers/main_classes/tokenizer`
[8]`https://huggingface.co/docs/transformers/model_doc/bert`

The classic learning algorithms were implemented with the help of *Scikit-Learn* library [38], the deep learning algorithms with *Keras* [12], and the BERT classifier with PyTorch [37].

3.5. Hyperparameter Optimization

Hyperparameter optimization or tuning is the step of choosing the optimal parameters for a classifier such as to minimize the generalization error [5]. Among various alternatives that could be considered, like exhaustive grid search, random search [6] or Bayesian optimization [44], we opted for evolutionary optimization (EO).

Evolutionary optimization using population-based probabilistic search algorithms [39] could drastically speed up the hyperparameter optimization while producing a good-enough combination of parameter values. Noticing the vast literature accompanying the metaheuristic design of DNNs [35] or recent applications of DL where parameters were selected with the help of genetic algorithms [19, 46] or suggestions that EO could outperform Bayesian optimization [32], we decided to employ a classical genetic algorithm for hyperparameter search.

We used Sklearn-genetic-opt library[9][3] for implementing the GA-based EO. Sklearn-genetic-opt makes usage of the Deap framework[10] [17], which supplies many evolutionary algorithms needed for solving optimization problems.

The GA was designed as following: given a number *n* of parameters to optimize for some specific classifier, a chromosome is a vector $(P_1, P_2, ..., P_n)$ of values selected for each parameter. A population consisting of 10 individuals which are evolved over 20 generations, with a crossover probability of 0.8 and mutation probability to 0.1. Individuals are selected for the next generation with a standard elitist tournament of size 3. Internally, each individual is evaluated using the accuracy as fitness function, computed with 3-folds cross-validation.

In the case of the classic ML algorithms all the parameters described in the official Sklearn documentation were optimized. In the case of the LSTM and CNN, we considered among the parameters the following: the network capacity (the number of hidden layers and the number of units per layer), the activation function, the regularization function, drop-out rate. Because both CNN and LSTM need the embedding weight parameter which is 2D tensor, we modified the source code of Sklearn-genetic-opt in order to transmit the multi-dimmensional parameters directly to Deap. Other additional parameters important for DL were optimized: batch size, number of epochs, initializer functions.

In general, convergence can be seen after 10-15 generations, thus evolving the populations over 20 generations is enough to guarantee a good parameter selection. In table 1 we present the parameters and their optimum values for each classifier used in our experiments.

In the case of BERT classifier, training and testing just one model is extremely time consuming therefore we decided not to perform evolutionary optimization. Instead of cross-validation, 10% of the training data was used for validation. The model was trained for 4 epochs because after this number the accuracy on the validation set started to decrease.

4. Experiments and Results

As mentioned in subsection 3.1, we worked with the *News Category* dataset. Details about the dataset are presented in subsection 3.1. We applied all the preprocessing steps as described in subsection 3.2 and we considered two different feature extraction methods, as indicated in subsection 3.3. For each encoding we selected the classifiers indicated in subsection 3.4 and we trained them using the best parameters found during the hyperparameter optimization process described in subsection 3.5 to obtain relevant models for topic classification.

[9]`https://sklearn-genetic-opt.readthedocs.io/`
[10]`https://github.com/deap/deap`

Table 1. Optimal parameters identified with EO hyperparameter search

Classifier	Optimal parameters (parameter_name=parameter_value)
Bernoulli NB	alpha=0.222, binarize=0.07, fit_prior=True
Random Forest	n_estimators=23, criterion=entropy, max_depth=None, min_sample_split=18, min_sample_leaf=8, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.01, bootstrap=True, oob_score=True, warm_start=False, class_weight=None, ccp_alpha=0.005
Linear SVM	dual=False, C=0.233, penalty=l2, fit_intercept=True, intercept_scaling=2.575, class_weight=None, tol=0.0001, loss=squared_hinge, multiclass=ovr, max_iter=431
LSTM	batch_size=559, epochs=16, activation=tanh, recurrent_activation=sigmoid, kernel_initializer=lecun_normal, recurrent_initializer=he_uniform, bias_initializer=zeros, unroll=False, kernel_regularizer=None, bias_regularizer=None, activity_regularizer=None, use_bias=True, recurrent_regularizer=l1, mask_zero=False, optimizer=rmsprop, loss=sparse_categorical_crossentropy, dropout_rate=0.484, embedding_layer_size=300, n_hidden_layers=2, first_hidden_layer_size=89, second_hidden_layer_size=63
CNN	batch_size=880, epochs=17, activation=softsign, kernel_initializer=orthogonal, use_bias=True, bias_initializer=glorot_normal, kernel_regularizer=None, bias_regularizer=l1, activity_regularizer=l2, optimizer=adamax, mask_zero=True, kernel_size=3, padding=same, pool_size=2, pool_strides=1, loss=sparse_categorical_crossentropy, dropout_rate=0.264, embedding_layer_size=300, n_hidden_layers=1, hidden_layer_size=112

The optimization and training was done on ≈ 151000 instances (75% of the dataset) while the testing on ≈ 51000 (25% of the dataset). We collected the following performance metrics for each model:

- Top-1, Top-2 and Top-3 accuracy on the test set (%),
- execution time of the hyperparameter optimization process $\rightarrow$ opt (s),
- training time of the classifier with optimum parameters $\rightarrow$ train (s), and
- classifier prediction time on the test set $\rightarrow$ test (s).

The results of each classifier are presented in Table 2.

Table 2. Classification performances

Classifier	Feature Extraction	Top-1 Acc. (%)	Top-2 Acc. (%)	Top-3 Acc. (%)	opt (s)	train (s)	test (s)
Bernoulli NB	TFIDF	64.2	78.9	85.2	443	0.66	0.04
Random Forest	TFIDF	31.0	40.7	50.2	2992	14.5	0.1
Linear SVM	TFIDF	68.0	81.8	87.1	8005	25.8	0.03
LSTM	Word2vec	67.6	80.5	85.6	286062	130.2	9.5
CNN	Word2Vec	66.2	80.1	85.3	37317	46.5	1.5
BERT		75.52	88.13	92.3		11189	357

As expected, BERT achieved the best classification accuracy in all three cases. The Top-1, Top-2, and Top-3 accuracies are approximately 7.5%, 6%, and 5% higher when compared to the second best algorithm, Linear SVM. But all of this came with a very high computational cost. Fine tuning a single BERT classifier over 4 epochs was approx. 430 times slower when compared to Linear SVM. The difference on the test data is even bigger, BERT taking around 350 seconds while Linear SVM taking less than 1 second. Because training a single BERT model took around 3 hours, hyperparameter optimization could not be performed on this classifier. But, as we noticed, this process is not needed because just one BERT classifier trained with the recommended parameters supplies state-of-the-art results.

The third best algorithm, LSTM, achived similar accuracies to SVM in the Top-1 and Top-2 cases but underperformed more significantly in the Top-3 case. In terms of accuracy CNN performed slightly worse than LSTM but the difference can be considered negligible. Bernoulli NB achieved the second worse Top-1 accuracy score, of around 64%, but had a comparable performance to CNN and LSTM in the Top-2 and Top-3 cases. The Random Forest classifier had by far the worst accuracy, even the Top-3 accuracy being considerably lower than the Top-1 accuracy of the Bernoulli NB.

We took advantage of a very powerful machine to run all experiments. Even so, time spent for hyperparameter optimization and model training, with the exception of Bernoulli NB, are not negligible (especially for BERT and the deep learning algorithms). The optimization time for Bernoulli was around 7 minutes while for linear SVM was almost 50 minutes. Searching for the best structure and parameters for CNN took $\approx$ 10 hours and for LSTM $\approx$ 79 hours.

Regarding the training time, Bernoulli NB was by far the faster with a training time of under 1 second. This was excepted due to the simplistic nature of the algorithm. Random Forest and Linear SVM had the second and third best training times of around 15 seconds and 25 seconds, respectively. Compared to Linear SVM, CNN took about double the time to train while LSTM was 5 times slower. Regarding the testing times, all the classic algorithms performed extremely well, having execution times of under 0.2 seconds. CNN had a testing time of around 1.5 seconds while LSTM of around 9.5. Although considerably slower than the classic algorithms, these times are very good considering that the testing was made on over 50000 records. BERT classifier had by far the slowest training and testing times.

If no a-priori English language knowledge in the form of pre-trained Word2Vec embeddings like Glove [40] or the deployment of a BERT [14] model is infeasible due to hardware limitations, we shall note that SVM remains the best option. Probably better results could be achieved with LSTM or CNN with pre-trained embeddings, but with an additional computational cost: number of weights to learn being increased due to the size of the pre-trained language models.

5. Conclusions

Within the larger scope of a media surveillance project [13] we constructed a system capable of inferring the topic discussed within short texts, which are common within social media platforms. Due to the lack of a publicly available social media dataset with manually labeled topic classes, we identified an appropriate candidate which can be used for training machine learning models. After augmenting the quality of the selected public dataset we inspected the average text lengths and conclude that these are similar with those usually found in microblogging.

After processing the data following the standard recommended methodology found in the literature, we built and evaluated various models constructed using standard machine leaning , deep learning, and fine-tuned the state-of-the-art BERT model for the topic classification task. We consider that the 26 topic classes used in our work should be enough for any general topic analysis task. Given the big number of parameters to optimize for the classic and deep learning models, we opted to perform hyperparameter search using the evolutionary optimization approach.

Considering the limitations imposed on our project, we found that the SVM with a linear kernel is the most robust one, and TF-IDF encoding is sufficient if no additional linguistic resources are available. Using LSTM with Word2Vec embeddings achieves a similar accuracy to linear SVM but with a significantly increased computational cost. As indicated by the literature, Bernoulli Naive Bayes gives robust results, but Random Forest does not perform well, having a Top-1 accuracy of 31% which is only around 3-4% above what ZeroR classification would achieve.

If one disposes plenty of available resources, and processing times are not a barrier, then a fine-tuned BERT classifier will deliver the best classification performance.

Acknowledgement

This paper was financed by the project with the title “*Platformă inovativă pentru măsurarea audienţei TV, identificarea automată a telespectatorilor şi corelarea cu date analitice din platforme de socializare online*” (Innovative platform for measuring TV audience, automatic identification of viewers and correlating it with analytic data from social media). The project was cofinanced by "Fondul European de Dezvoltare Regională prin Programul Operaţional Competitivitate (POC) 2014-2020, Axa prioritară: 2-Tehnologia Informației şi Comunicaţiilor (TIC) pentru o economie digitală competitivă." (the European Regional Development Fund (ERDF) through the Competitiveness Operational Program 2014-2020, Priority Axis 2 - Information and Communication Technology (ICT) for a competitive digital economy), project code SMIS 2014+:128960, beneficiary: CICADA TECHNOLOGIES S.R.L. The project is part of the call: POC/524/2/2/ “Sprijinirea creşterii valorii adăugate generate de sectorul TIC şi a inovării în domeniu prin dezvoltarea de clustere” (Supporting the added value generated by the ICT sector and innovation in the field through cluster development). The content of this material does not necessarily represent the official position of the European Union or the Romanian Government.

References

1. Agrawal, A., Fu, W., Menzies, T.: What is wrong with topic modeling? and how to fix it using search-based software engineering. Information and Software Technology 98, pp. 74–88 (2018)
2. Albanese, F., Feuerstein, E.: Improved topic modeling in twitter through community pooling. In: String Processing and Information Retrieval - 28th International Symposium, SPIRE 2021. LNCS, vol. 12944, pp. 209–216. Springer (2021), `https://doi.org/10.1007/978-3-030-86692-1_17`
3. Arenas Gomez, R.: GASearchCV - sklearn genetic opt 0.4.0 documentation (2021), `https://sklearn-genetic-opt.readthedocs.io/en/0.4.0/api/gasearchcv.html`
4. Beel, J., Gipp, B., Langer, S., Breitinger, C.: Research-paper recommender systems: A literature survey. International Journal on Digital Libraries 17(4), pp. 305–338 (2016)
5. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems 24: 25th Annual Conference on NIPS 2011. Proceedings. pp. 2546–2554 (2011)
6. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research 13, pp. 281–305 (2012)
7. Blei, D., Lafferty, J.: Correlated topic models. Advances in neural information processing systems 18, pp. 147 (2006)
8. Blei, D.M.: Probabilistic topic models. Communications of the ACM 55(4), pp. 77–84 (2012)
9. Boyd-Graber, J.L., Blei, D.M.: Syntactic topic models. In: Proc. of the 22nd Annual Conf. on Neural Information Processing Systems, 2008. pp. 185–192 (2008)
10. Breiman, L.: Random forests. Machine Learning 45(1), pp. 5–32 (2001)
11. Cheng, X., Yan, X., Lan, Y., Guo, J.: Btm: Topic modeling over short texts. IEEE Transactions on Knowledge and Data Engineering 26(12), pp. 2928–2941 (2014)
12. Chollet, F., et al.: Keras. `https://keras.io` (2015)

13. Cicada Technologies: Innovative platform for measuring tv audience, automatic identification of viewers and correlating it with analytic data from social media (2020), https://www.cicadatech.eu/projects/, accessed on April 08, 2022
14. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Volume 1 (Long and Short Papers). pp. 4171–4186. ACL (2019), `https://doi.org/10.18653/v1/n19-1423`
15. Eisenstein, J.: What to do about bad language on the Internet. In: Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, 2013. pp. 359–369. ACL (2013)
16. Fan, X., Lin, H., Yang, L., Diao, Y., Shen, C., Chu, Y., Zou, Y.: Humor detection via an internal and external neural network. Neurocomputing 394, pp. 105–111 (2020)
17. Fortin, F.A. et al.: DEAP: Evolutionary algorithms made easy. Journal of Machine Learning Research 13, pp. 2171–2175 (2012)
18. Gentzkow, M., Kelly, B., Taddy, M.: Text as data. Journal of Economic Literature 57(3), pp. 535–74 (September 2019)
19. Gorgolis, N., Hatzilygeroudis, I., Istenes, Z., Gyenne, L.: Hyperparameter optimization of LSTM network models through genetic algorithm. In: 10th Intl. Conf. on Information, Intelligence, Systems and Applications, IISA 2019. pp. 1–4. IEEE (2019)
20. Gupta, M.R., Bengio, S., Weston, J.: Training highly multiclass classifiers. Journal of Machine Learning Research 15(1), pp. 1461–1492 (2014)
21. Guzella, T.S., Caminhas, W.M.: A review of machine learning approaches to spam filtering. Expert Systems with Applications 36(7), pp. 10206–10222 (2009), `https://doi.org/10.1016/j.eswa.2009.02.037`
22. Hong, L., Davison, B.D.: Empirical study of topic modeling in Twitter. In: Proc. of the 3rd Workshop on Social Network Mining and Analysis, SNAKDD 2009 . pp. 80–88. ACM (2010), `https://doi.org/10.1145/1964858.1964870`
23. Honnibal, M., Johnson, M.: An improved non-monotonic transition system for dependency parsing. In: Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015. pp. 1373–1378. ACL (2015)
24. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Reading text in the wild with convolutional neural networks. International journal of computer vision 116(1), pp. 1–20 (2016)
25. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Proceedings of ECML-98, 10th European Conference on Machine Learning. LNCS, vol. 1398, pp. 137–142. Springer (1998)
26. Lee, K. et al.: Twitter trending topic classification. In: 2011 IEEE 11th Intl. Conf. on Data Mining Workshops (ICDMW), Proceedings. pp. 251–258. IEEE Computer Society (2011)
27. Liu, B.: Sentiment analysis: Mining opinions, sentiments, and emotions. Cambridge University Press (2020)
28. McCallum, A., Nigam, K.: A comparison of event models for naive bayes text classification. In: Learning for Text Categorization: Papers from the 1998 AAAI Workshop. pp. 41–48 (1998)
29. Medhat, W., Hassan, A., Korashy, H.: Sentiment analysis algorithms and applications: A survey. Ain Shams engineering journal 5(4), pp. 1093–1113 (2014)
30. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings. pp. 3111–3119 (2013)

31. Misra, R.: News Category Dataset - Sculpturing Data for ML (June 2018), http://doi.org/10.13140/RG.2.2.20331.18729
32. Mori, N., Takeda, M., Matsumoto, K.: A comparison study between genetic algorithms and bayesian optimize algorithms by novel indices. In: Proc. of the 7th Annual Conf. on Genetic and Evolutionary Computation. pp. 1485–1492. ACM (2005)
33. Müller, T., Cotterell, R., Fraser, A.M., Schütze, H.: Joint lemmatization and morphological tagging with lemming. In: Proc. of the 2015 Conf. on Empirical Methods in Natural Language Processing, EMNLP 2015. pp. 2268–2274. ACL (2015)
34. Oh, S.: Top-k hierarchical classification. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence. pp. 2450–2456. AAAI Press (2017)
35. Ojha, V.K., Abraham, A., Snásel, V.: Metaheuristic design of feedforward neural networks: A review of two decades of research. Engineering Applications of Artificial Intelligence 60, pp. 97–116 (2017)
36. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: Proc. of the 30th Intl. Conf. on Machine Learning, ICML 2013. JMLR Workshop and Conference Proc., vol. 28, pp. 1310–1318. JMLR.org (2013)
37. Paszke, A. et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019)
38. Pedregosa, F. et al.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, pp. 2825–2830 (2011)
39. Pelikan, M., Goldberg, D.E., Lobo, F.G.: A survey of optimization by building and using probabilistic models. Computational Optimizations and Applications 21(1), pp. 5–20 (2002)
40. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing, EMNLP 2014. pp. 1532–1543. ACL (2014)
41. Rahman, M.A., Akter, Y.A.: Topic classification from text using decision tree, K-NN and Multinomial Naïve Bayes. In: 2019 1st Intl. Conf. on Advances in Science, Engineering and Robotics Technology (ICASERT). pp. 1–4. IEEE Press (2019)
42. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proc. of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA, Malta (May 2010)
43. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information processing and management 24(5), pp. 513–523 (1988)
44. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: 26th Annual Conference on NIPS 2012. Proceedings. pp. 2960–2968 (2012)
45. Vayansky, I., Kumar, S.A.: A review of topic modeling methods. Information Systems 94, pp. 101582 (2020)
46. Violos, J., Tsanakas, S., Androutsopoulou, M., Palaiokrassas, G., Varvarigou, T.: Next position prediction using lstm neural networks. In: 11th Hellenic Conference on Artificial Intelligence. p. 232–240. ACM (2020)
47. Wang, X., McCallum, A.: Topics over time: a non-Markov continuous-time model of topical trends. In: Proc. of the 12th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, 2006. pp. 424–433. ACM (2006)
48. Zeng, J., Li, J., Song, Y., Gao, C., Lyu, M.R., King, I.: Topic memory networks for short text classification. In: Proc. of the 2018 Conf. on Empirical Methods in Natural Language Processing. pp. 3120–3131. ACL (2018)