TextBrew: Automated Model Selection and Hyperparameter Optimization for Text Classification

Rushil Desai[1], Aditya Shah[2], Shourya Kothari[3], Aishwarya Surve[4] and Dr. Narendra Shekokar[5]
Department of Computer Engineering, Dwarkadas J. College of Engineering
Mumbai, India

***Abstract*—In building a machine learning solution, algorithm selection and hyperparameter tuning is the most time-consuming task. Automated Machine Learning is a solution to fully automate the process of finding the best model for a given task without actually having to try various models. This paper introduces a new AutoML system, TextBrew, explicitly built for the NLP task of text classification. Our system provides an automated method for selecting transformer models, tuning hyperparameters, and combining the best models into one by ensembling. Keeping in mind that new state-of-the-art models are being constantly introduced, TextBrew has been designed to be highly flexible and thus can support additional models easily. In our work, we experiment with multiple transformer models, each with numerous different hyperparameter settings, and select the most robust models. These models are then trained on multiple datasets to obtain accuracy scores, which are then used to build the meta-dataset to train the meta-model. Since text classification datasets are not as abundant, our system generates synthetic data to augment the meta-dataset using CopulaGAN, a deep generative model. The meta-model is an ensemble of five models, which predicts the best candidate model with an accuracy of 78.75%. The final model returned to the user is an ensemble of all the best models that can be trained under the given time constraint. Experiments on various datasets and comparisons with existing systems demonstrate the effectiveness of our system.**

***Keywords*—Automated machine learning; AutoML; NLP; transformer models; hyperparameter optimization; CopulaGAN; generative model; meta-learning**

I. Introduction

As more and more people recognize the actual value of data and try to get the most out of it, the demand for machine learning tools is increasing. But the complexity of the tasks involved in machine learning can be overwhelming for non-ML experts, and this is where automated machine learning comes into the picture. Non-ML experts can use AutoML for building ML projects. ML experts can also use it to perform repetitive tasks to save time and accelerate machine learning research by automating the development of models. Using AutoML, one can train high-performing models without worrying about hyperparameters, model architecture, or cross-validation strategies [1]. It aims to automate the model selection process, as shown in [2], [3], and its applications are increasing by the day, hand-in-hand with the growth of applications of machine learning.

Machines must use complex data processing and state-of-the-art machine learning algorithms to work with natural language. It is also essential to decide the particular machine learning method based on the dataset and the task. Selecting the correct method may become difficult as more accurate methods are constantly being developed. Even after considering all these factors for choosing the algorithms and methods, getting the best result is not assured. AutoML for text classification (TextBrew) can address all these challenges.

Finding the best model and the best set of hyperparameters is tedious. This task requires a lot of time and resources since training a model on a dataset may take hours or even days, subject to the size of the dataset and the model being used, and then changing hyperparameters and training the dataset all over again becomes an even more tedious and patience-testing task. We aim to tackle this issue and give the user the best model for an NLP classification task.

This paper presents a system that returns the trained model predicted to be the best for it within a particular time limit when given a dataset as an input. Our system is such that it is effortless to add new state-of-the-art transformer models to the pre-existing pool of models for the system to consider those models. In this paper, we have performed all experiments on BERT [4], ALBERT [5], and XLNet [6] for text classification using multiple hyperparameter combinations for each. A major part of building our AutoML system is creating the meta-dataset, which is made by training and testing several deep learning models on multiple text classification datasets. This process has been made efficient using generative models to synthesize data instead of running models on many datasets. The results demonstrate the effectiveness of our system.

A. Contribution and Organization of the Paper

To sum up, we aim to reduce the time spent by a data scientist on selecting a particular model and its hyperparameter values. As seen in the next section, a significant drawback of many AutoML systems is the inability to support additional, more complex models. TextBrew has been developed keeping this problem in mind. The contributions of our paper are as follows:

1) We show the effectiveness of using deep generative models to synthesize the training data instead of spending hours running models on a large number of datasets.
2) TextBrew is a flexible system that can accommodate the new models with the least number of changes possible as it is not built around any specific set of models. All that is needed for the inclusion of a new model is for it to be added to the model pipeline which is run while generating the meta-dataset.

This paper is structured as follows. In the next section, we review the related literature. Section III discusses the

methodology we have adopted. The results are described in Section IV, followed by the discussion in Section V. Sections VI and VII outline the conclusion and a plan for future steps, respectively.

II. Review of Literature

AutoML is a relatively new research topic and has made significant progress in recent years. Many surveys [7] summarize the works of other researchers on this topic. Most of these works focus on individual modules such as neural architecture search (NAS) or hyperparameter optimization (HPO) in classifying tabular data using classical algorithms. There is significantly less work done on AutoML for text classification, which is quite different from classifying tabular data because textual data is unstructured and requires more computationally intensive algorithms.

There are works published that compare the performance of various existing AutoML tools on text classification tasks. Blohm et al. [8] attempt to answer the question of whether AutoML can be effectively applied to text classification or not. They compare the performance of four AutoML tools on thirteen text classification datasets and document how they perform as opposed to human-engineered models. Their experiments show that AutoML systems perform better than humans on 4 out of 13 tasks, and this number will continue increasing as more sophisticated AutoML tools for NLP tasks are developed.

Some individual works have tackled the problem of AutoML for text classification. Madrid et al. [9] propose a method that automatically builds text classification pipelines based on the metadata obtained by running experiments using standard algorithms on 81 datasets. While it is effective, metadata obtained from 81 datasets could be insufficient for a machine learning model to learn and give optimal results.

Wong et al. [10] incorporate transfer learning to reduce the computational cost of Neural AutoML. Their system learns the hyperparameter choices common to multiple tasks and uses this to accelerate the network design for a new task. The results show a reduction in convergence time for text classification. The drawback of this system is that meta-overfitting has not been dealt with, which is an issue.

Gomez et al. [11] have used a hyper-heuristic approach by employing a genetic algorithm to evolve a population of meta-rules to decide the best model for the particular text classification dataset. This work considers only simple models: K-Nearest Neighbor, Logistic Regression, Support Vector Machine, and Multinomial Naive Bayes. While this approach is interesting, it is not ideal because an AutoML system for the task of text classification needs to be able to handle the high computational cost and complexity of state-of-the-art transformer models.

A similar issue is present in the system proposed by Feurer et al. [12]. They use Bayesian optimization to capture the relationship between the hyperparameter values and the model performance. This system includes 15 classifiers, 14 feature preprocessing methods, and 4 data preprocessing method in its search space. Including deep learning methods would have made their system more competitive with other AutoML systems for text classification.

Additionally, there exist some open-source AutoML systems that work with text data. Jin et al. [13] propose the AutoML system AutoKeras, which efficiently performs neural architecture search using Bayesian optimization, enabling it to select the most profitable operation each time. Shi, Mueller, et al. [14] in their paper use transformer networks and stack ensemble them with classical tabular models to handle data that contain text, numeric, and categorical features. A drawback of this system is that it works with a limited search space compared to other AutoML tools.

We propose TextBrew, a simpler system for AutoML for text classification, which offers an automated way for selecting the best transformer models and then training these models on the dataset with hyperparameter tuning within the time constraints provided by the user. The issue of creating a large meta-dataset, which takes a tremendous amount of time and effort, is solved by using GANs [15] to synthesize additional data.

III. Methodology

A. Overview

The system offers an automated way for selecting the best transformer models and then training these models on the dataset with the proper hyperparameter settings. It is designed in such a manner that it can handle both binary as well as multiclass prediction datasets without explicitly stating so. As shown in Fig. 1, this system for automated text classification can be segregated into two parts:

1) Building and Training the Meta-model.
2) Using TextBrew for Model Prediction.

Meta-model refers to the model produced on training the meta-learning algorithm. In the first part, “Building and Training the Meta-model”, top-performing transformer models with many different combinations of hyperparameters are considered. After conducting experiments, the best-performing candidate models are shortlisted. “Candidate model” is used in this paper to refer to a model with a particular combination of hyperparameter values. Next, the multinomial Naive Bayes model is trained along with the selected candidate models on a collection of text classification datasets. Due to the small size of the meta-dataset (the dataset on which the meta-model is trained) generated at this stage, CopulaGAN [16] (a deep generative model) is used to synthesize additional data, helping our meta-model train better. From the meta-dataset, we use the accuracy of the Naive Bayes and ALBERT_2 models along with the dataset size as the predictor variables to predict the performance of other models. A soft voting ensemble consisting of five models was used for building the meta-model, and this meta-model was trained on the generated dataset to achieve better accuracy.

In the second part, “Using TextBrew for Model Prediction”, the meta-model is used for model prediction. The best model predicted by the meta-model is trained on the dataset provided by the user under the given time constraint. After training the best candidate model predicted by the meta-model, the next best predicted model is trained if time remains. This is continued until the specified time is exceeded. Finally, an ensemble of all these models is returned to the user.

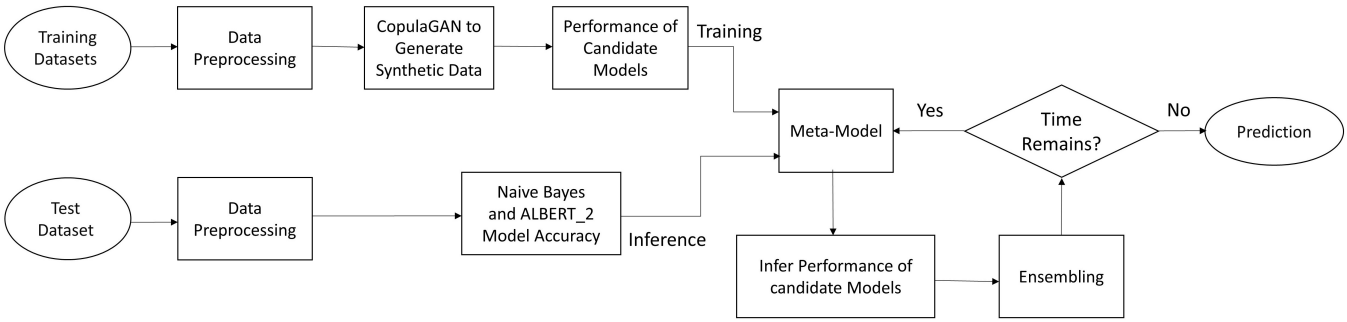

Fig. 1. Architecture Diagram of the TextBrew System.

B. Building and Training the Meta-Model

Here, we describe how the meta-dataset is created and how the meta-model is trained.

For the real-world usage of this system, the system needs to be able to take into consideration a lot of different models. Transformer models are state-of-the-art machine learning models for NLP text classification tasks. There are a lot of different choices available, with GPT [17], BERT [4], RoBERTa [18], ELECTRA [19], ALBERT [5] and BART [20] being just a few of them. In fact, several different pre-trained models are based on each of these language models. For instance, BanglaBert [21] is a BERT-based Natural Language Understanding model pre-trained in Bangla. In addition, new models are being built at a blazing pace, and state-of-the-art models are being replaced frequently. This begs for a flexible system to accommodate the new models with the least number of changes possible.

1) Candidate Model Selection: Since transformer models are computationally costly, we perform all experiments needed to design this system using three models. The system is independent of the model choices and count. One can add more models in the same way described in this paper.

Gasparetto et al. [22], in their work, tested many algorithms on multiple text classification datasets, and it can be seen that these XLNet and BERT-base are among the top performing models in terms of accuracy.

In addition to these two models, we select ALBERT (A Light BERT) as one of our models as it has an architecture similar to that of BERT but has a fraction of the total number of parameters [5]. This makes it fast and less computationally expensive and so can help save time. Above are the reasons for selecting XLNet, BERT-base, and ALBERT for our experiments. These models are among the best for text classification.

Since all three transformer models have multiple hyperparameters that need to be tuned to an appropriate value, there are effectively many candidate models from which to choose. So, the next step is selecting the hyperparameter values for each model. We experimented with different combinations of hyperparameter values and picked the ones that gave us the best results when trained on multiple datasets.

The selected possible values of the hyperparameters for the models are shown in Table I. A short description of the hyperparameters is given below:

TABLE I. HYPERPARAMETERS AND SELECTED VALUES

Hyperparameters	Selected values
num_warmup_steps	[0, 1]
init_lr	[2e-5, 1e-5]
adam β_1	[0.8, 0.9]
adam β_2	[0.9, 0.999]
powers	[1, 1.5]
epochs	[3, 5]

1) Num_warmup_steps: It is a parameter used to lower the learning rate to reduce the impact of deviating the model from learning on sudden new data set exposure.
2) Init_lr: It is the initial learning rate before training and after warm-up steps.
3) Adam β_1, Adam β_2: The hyper-parameters β_1 and β_2 of Adam are initial decay rates used when estimating the moments of the gradient, which are multiplied by themselves at the end of each training step.
4) Powers: The power to use for polynomial decay.
5) Epochs: The number of passes or iterations of the training dataset by the machine learning model.

We choose the best set of hyperparameters by using an algorithm similar to grid-search. We run the transformer models with each possible combination of the hyperparameter values and compute the set of hyperparameters for which the model returns the best accuracy as output. In this case, each model effectively has 2^6 candidate models, counting all combinations of the six hyperparameters mentioned above.

Next, we selected three datasets to reduce bias in training the models. The three datasets are Sarcasm Detection [23], E-Mail classification NLP [24] and Financial phrase-bank [25]. Before training the candidate models on these datasets, the datasets were passed through a standard preprocessing pipeline. First, the text is converted to lowercase, and then we remove the URLs, non-ASCII characters, punctuations, and stopwords from the text. As this paper focuses mainly on model selection and hyperparameter optimization and not on the preprocessing of the data, the same preprocessing pipeline is used to clean all the datasets which are passed through the system.

All the candidate models for the three models, XLNet, BERT-base, and ALBERT, were then trained on the preprocessed datasets. Our system records how each hyperparameter combination performs on each dataset. Taking every possible combination of the hyperparameter values, we get 64 possible outcomes for each model. Hence, on trying these 64 outcomes for each of the three models for each of the three datasets, we effectively train 576 candidate models. The system records the accuracy attained by that particular model and the time required to train the model. From this dataset, we select the top six models (two from each of the three transformer models) as the shortlisted candidate models. We did this by averaging the accuracy of each unique hyperparameter combination model across the three datasets and taking the top two best-performing candidate models. This handles the situation where a model performs exceptionally well on one dataset and poorly on another. The top 6 models and the corresponding hyperparameter values are mentioned in Table II.

TABLE II. TOP SIX MODELS AND THEIR CORRESPONDING HYPERPARAMETERS

Model Name	Hyperparameters				
	epochs	*warmup_steps*	*learning_rate*	*adam* β_1	*adam* β_2
BERT_1	5	0	2e-5	0.9	0.999
BERT_2	5	0	1e-5	0.9	0.9
ALBERT_1	5	1	2e-5	0.8	0.9
ALBERT_2	3	0	1e-5	0.9	0.9
XLNet_1	3	1	1e-5	0.8	0.9
XLNet_2	3	1	2e-5	0.8	0.999

2) Meta-Dataset Preparation: After selecting the top six models, we create the meta-dataset for training the meta-model. We curated a list of 44 datasets consisting of both binary and multiclass text classification datasets. A longer list of datasets would have been beneficial, but for the task of English language text classification, the datasets publicly available are limited in number. On the other hand, a longer list of datasets means a lot more time to build the meta-dataset as more model training would have to be done. We solve this problem by synthesizing data using GANs, as described later in the paper. For each dataset, our system automatically trains all the selected candidate models and one additional multinomial Naive Bayes model. The accuracy and the training time taken for all these models are recorded.

The pipeline for training the multinomial Naive Bayes model consists of a count vectorizer that transforms the text into a vector based on the frequency of each word. A count matrix is created in which a column of the matrix represents each unique word in the dataset, and each row of the dataset is a row in the matrix, and the cells contain the count of the word in that text. Then, we transform the count matrix into a normalized TF-IDF representation. We use TF-IDF instead of the raw frequencies to reduce the impact of the frequently occurring words that are less informative than other words that occur in a small fraction of the dataset. We then train the multinomial Naive Bayes classifier on this. The reason for considering Naive Bayes for our model prediction is stated in the next section.

Table III shows a few records from our final meta-dataset.

3) Feature Extraction: The prerequisite step for training a classifier is feature extraction. Text embedding is the most popular feature extraction method for text-oriented tasks [29]. Calculating embeddings for entire datasets is not feasible because of the computational expense that comes with it. Since the performance of any model depends on the dataset it has been trained on, the performance metrics of a model can be treated as a feature that is representative of the dataset. This approach is the basis for feature extraction in this paper.

Above is the reason for including multinomial Naive Bayes in the list of models. The accuracy of the Naive Bayes model is part of the feature space. The reason for choosing Naive Bayes is that it is extremely fast to train on the datasets compared to other models, making it efficient for use as a feature extraction tool. We also use the number of records in the dataset to predict the accuracy values of various models. Further, since ALBERT is A Light BERT model, it is faster to train than other transformer models we have chosen. Studying Table IV, we see that ALBERT_2 consistently takes the least time to train. This makes it possible for us to take ALBERT_2 accuracy scores as the third predictor variable to expand the feature space and, in turn, increase the accuracy of our meta-model. After running experiments to validate our hypothesis, we see that using the performance of the ALBERT_2 model along with Naive Bayes performance and the dataset size improves the accuracy of the meta-model for predicting the best model. Adding the ALBERT_2 accuracy score to the feature space boosts the accuracy by 26%, from 0.625 to 0.788, as shown in Fig. 2.

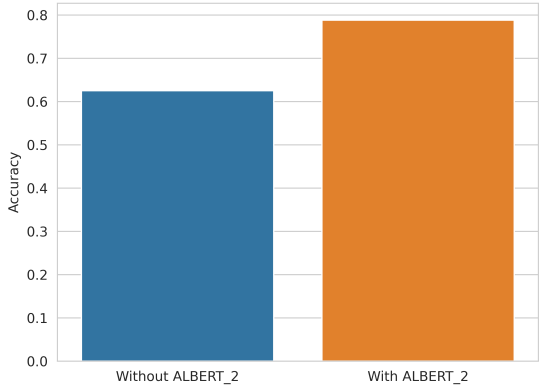

Fig. 2. Improvement in Overall Accuracy after Adding ALBERT_2 Accuracy Score to the Feature Space.

4) Using Generative Models to Synthesize Data: Creating an extensive dataset requires a lot of time, and even if time constraints were not to be considered, publicly available text classification datasets are limited in number. Furthermore, overfitting is a problem faced by a model when the model predicts exceedingly well on the training dataset but cannot provide similar results on the testing dataset. It is known that small datasets are prone to overfitting. As a result, we need to tackle the overfitting problem caused by the small size of the dataset.

In this paper, we use GANs [15] to synthesize data and increase the dataset size to obtain a better result from the meta-classifier. Generative adversarial networks are an unsupervised learning approach to generative modeling using deep learning methods such as convolutional neural networks (CNN). Generative modeling involves automatic learning of patterns and the

TABLE III. CANDIDATE MODELS AND THEIR ACCURACIES ON DIFFERENT DATASETS

Dataset	Naive Bayes	BERT_1	BERT_2	ALBERT_1	ALBERT_2	XLNet_1	XLNet_2	Best Model
Cyberbullying [26]	0.8885	0.9925	0.9925	0.9950	0.9925	0.9925	0.9943	**ALBERT_1**
WELFake [27]	0.8510	0.9657	0.9531	0.9743	0.9788	0.9748	0.9834	**XLNet_2**
Amazon [28]	0.8550	0.9343	0.9086	0.9571	0.8200	0.6229	0.7257	**ALBERT_1**
Yelp [28]	0.7800	0.9486	0.8571	0.9371	0.8286	0.6514	0.7743	**BERT_1**
IMDb [28]	0.8000	0.9313	0.8053	0.9504	0.7939	0.6603	0.8397	**ALBERT_1**

TABLE IV. CANDIDATE MODELS AND THEIR TRAINING TIME (IN SECONDS) ON DIFFERENT DATASETS

Dataset	Naive Bayes	BERT_1	BERT_2	ALBERT_1	ALBERT_2	XLNet_1	XLNet_2	Fastest Model
Cyberbullying [26]	0.16	479.21	478.63	496.30	298.16	431.29	431.49	**ALBERT_2**
WELFake [27]	1.46	344.80	330.15	336.70	202.68	296.60	287.91	**ALBERT_2**
Amazon [28]	0.02	82.80	65.37	66.66	39.45	68.52	57.11	**ALBERT_2**
Yelp [28]	0.01	65.74	65.80	65.53	39.42	57.38	57.40	**ALBERT_2**
IMDb [28]	0.02	49.85	49.70	49.14	29.83	43.66	43.62	**ALBERT_2**

regularities of the data in a way that can be used to synthesize new examples that plausibly could have been drawn from the original dataset.

We employ the CopulaGAN model [16] to generate 150 data points for each group of datasets having a particular candidate model as the best model. This ensures that the resulting dataset is not unbalanced. The CopulaGAN model is a variation of the CTGAN model [30] which takes advantage of the CDF-based transformation that the GaussianCopulas apply to make the underlying CTGAN model task of learning the data easier. We make use of different metrics to compare the synthesized data and the original data. Seeing the results in Table V, we can be assured of the quality of the synthetic data.

TABLE V. METRICS FOR SYNTHETIC DATA EVALUATION

Kolmogorov–Smirnov Test	Multiclass Decision Tree Classifier
0.69	0.59

The Kolmogorov-Smirnov test [31] is based on the maximum difference between an empirical and a hypothetical cumulative distribution. The test concerns the agreement between the generated data and the original data. The Multiclass Decision Tree Classifier test lets the generated data pass through a decision tree generated on the original dataset. The resulting accuracy of this test indicates the percentage of data points that fall perfectly in accordance with the original data.

5) Meta-Model Details: We use a soft voting ensemble as our meta-model. This means that we predict the class with the largest summed probability from all the models that are part of the ensemble. Our ensemble consists of five models, each of which can perform a multiclass classification.

1) Multinomial Logistic Regression [32]
2) XGBClassifier [33]
3) C-Support Vector Classification following the One-vs-Rest scheme [34]
4) Random Forest Classifier [35]
5) AdaBoost Classifier [36]

We then use the generated data to train the meta-model. To see how the synthesized data improves the meta-model performance, we train it on the original data, record the accuracy, and then do the same on the synthetic data. As expected, we get a low accuracy of 0.475 on training the meta-model on a small dataset. This accuracy score is boosted to 0.7875 after using synthetic data, which consists of a lot more data.

C. Using TextBrew for Model Prediction

In this section, we describe how the user would use our proposed system and how the system makes use of the meta-model we trained in the previous section. The user passes a text classification dataset and an optional parameter: *allowed_training_time* as input to the system. The parameter *allowed_training_time* denotes the upper limit for the time the user expects the trained model to be returned as an output. This parameter helps users set a time limit for training the model if they have a time constraint. The default value of this parameter is set as 15 minutes in our experiments.

First, the dataset is passed through the same preprocessing pipeline we created for the datasets used for training, which is mentioned in the above sections. The preprocessing pipeline helps to clean the data and eliminate inconsistencies. The next step is to train Naive Bayes and ALBERT_2 models on the preprocessed dataset. We considered Naive Bayes because it is swift to train, as shown in Table IV. ALBERT_2 was considered because it was the fastest to train out of the six transformer models and could be used as a feature to predict the best model. The output of these two models, plus the size of the dataset, forms the feature space. These features are input to the meta-model, which predicts the best-performing model. The dataset given as input to the system is then trained on this predicted model, and the time taken for training is monitored. After training is completed, if the system has not exceeded the *allowed_training_time*, then it trains the next best model predicted on the same dataset. This continues until it exceeds the allowed time. Finally, an ensemble of all these trained models is created using a soft voting ensemble algorithm. The prediction probabilities for all the class labels are summed up, and the class label with the most considerable sum is selected as the predicted class. This ensemble is returned as the output.

IV. Results

As mentioned in the previous section, we developed a meta-model that correctly predicted the best model with an accuracy of 78.75% on using CopulaGAN to synthesize data, compared to 47.5% on using the original data. This is a strong score given that the meta-dataset is generated using only 44 datasets because of the limited availability of open-source datasets for text classification. Additionally, 95% of the time, the first model predicted is among the top three models in terms of accuracy. These numbers indicate that it is highly likely that the best model will be part of the ensemble returned by the system as it consists of the first few models predicted to perform well by the meta-model.

We also present the results of the experiments that were performed using different datasets and different existing AutoML solutions for text classification. FakeNews, Covid Tweets Sentiment Analysis, and Cyberbullying Classification are the datasets used. We compare our system with AutoKeras [13] and AutoGluon [14]. In order to create an unbiased environment for all tests, we use Intel® Xeon® 2.00GHz CPU, Tesla P100 16GB GPU, and 13 GB of free memory for our usage.

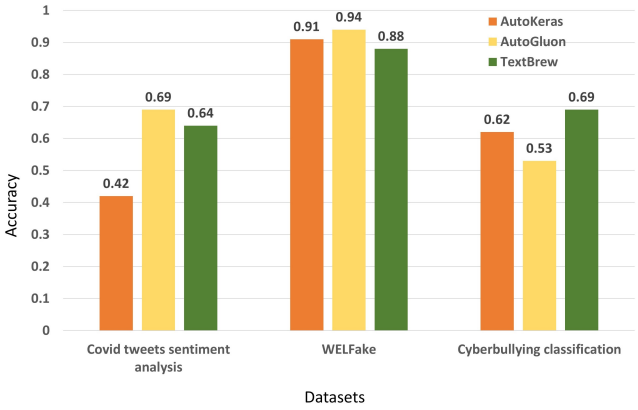

Fig. 3. Comparion of AutoML Systems, Including TextBrew (green), AutoKeras (orange) and AutoGluon (yellow).

We can see from Fig. 3 that our system gives the best accuracy on one dataset and a reasonable accuracy on the rest compared to AutoKeras and AutoGluon. Given that this is a prototype of our system and has only six candidate models originating from three different transformer models, i.e., BERT-base, XLNet, and ALBERT, the performance, as seen in Fig. 3, is extremely promising. Increasing the number of models described in the methodology section would lead to much better performance.

V. Discussion

Today, very few AutoML systems for the NLP task of text classification exist that incorporate state-of-the-art deep learning models. AutoGluon is one of the best-performing AutoML models for text classification, as seen in Fig. 3, and TextBrew follows closely behind. In fact, TextBrew beats AutoGluon on one out of the three datasets with an accuracy that is 16% higher. With access to more computing power, we can incorporate more state-of-the-art deep learning models and a more comprehensive range of hyperparameter options, which will enable TextBrew to be competitive and could challenge the best AutoML tools present today.

Moreover, TextBrew returns an ensemble of models that would give the best results on the given dataset. The best possible model out of the six selected models is found to be in the top three with an accuracy of 95%, of which 78.75% constitutes the situation where the first model predicted is actually the best. These numbers convey that the probability of the best candidate model being included in the ensemble generated by the system is very high.

The workings of our system back up the suggestion that automated machine learning does not have to be complicated to give good results. This is because this system does not employ any complicated methods or algorithms like genetic algorithms as seen in [11], deep reinforcement learning as seen in [10] and Bayesian optimization as visited in [12], [13]. The approach here is to build a meta-dataset and train classical ML models, which are used to predict the suitable transformer model. Nevertheless, TextBrew performs competitively considering the dataset's size and the approach's simplicity.

VI. Conclusion

This paper proposes TextBrew: an automated machine learning system for text classification. As discussed in the previous section, TextBrew suggests that AutoML systems do not have to be highly complex to perform well. The meta-model predicts the best possible model and suitable hyperparameters while considering the user's time constraint. Our meta-model predicts one of the top three models 95% of the time, with the best candidate model being predicted with an accuracy of 78.75%. This means that it is highly likely that the ensemble created by the system will include the best models.

The final model returned to the user is an ensemble of all the top candidate models that can be trained under the given time constraint. As seen in the results section, considering the low number of models in our system, TextBrew is promising, and it backs the idea that automated machine learning is not as complex as one would think.

VII. Future Work

For our future work, we aim to manually build and compile a greater number of text classification datasets to train our meta-model better. Additionally, we aim to access a more powerful computing system to include a much larger number of models and hyperparameter choices in our system.

References

[1] S. K. Karmaker ("Santu"), M. M. Hassan, M. J. Smith, L. Xu, C. Zhai, and K. Veeramachaneni, "Automl to date and beyond: Challenges and opportunities," *ACM Comput. Surv.*, vol. 54, no. 8, oct 2021. [Online]. Available: https://doi.org/10.1145/3470918

[2] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf

[3] N. Fusi, R. Sheth, and M. Elibol, "Probabilistic matrix factorization for automated machine learning," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 3352–3361.

[4] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[5] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," *CoRR*, vol. abs/1909.11942, 2019. [Online]. Available: http://arxiv.org/abs/1909.11942

[6] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *CoRR*, vol. abs/1906.08237, 2019. [Online]. Available: http://arxiv.org/abs/1906.08237

[7] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705120307516

[8] M. Blohm, M. Hanussek, and M. Kintz, "Leveraging automated machine learning for text classification: Evaluation of automl tools and comparison with human performance," pp. 1131–1136, 01 2021.

[9] J. Madrid, H. J. Escalante, and E. Morales, "Meta-learning of textual representations," 2019. [Online]. Available: https://arxiv.org/abs/1906.08934

[10] C. Wong, N. Houlsby, Y. Lu, and A. Gesmundo, "Transfer learning with neural automl," *Advances in neural information processing systems*, vol. 31, 2018.

[11] J. C. Gomez, S. Hoskens, and M.-F. Moens, "Evolutionary learning of meta-rules for text classification," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 131–132. [Online]. Available: https://doi.org/10.1145/3067695.3075601

[12] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 2755–2763.

[13] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 1946–1956.

[14] X. Shi, J. Mueller, N. Erickson, M. Li, and A. Smola, "Multimodal automl on structured tables with text fields," in *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[16] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Oct 2016, pp. 399–410.

[17] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[18] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[19] K. Clark, M. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: pre-training text encoders as discriminators rather than generators," *CoRR*, vol. abs/2003.10555, 2020. [Online]. Available: https://arxiv.org/abs/2003.10555

[20] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *CoRR*, vol. abs/1910.13461, 2019. [Online]. Available: http://arxiv.org/abs/1910.13461

[21] A. Bhattacharjee, T. Hasan, K. Samin, M. S. Rahman, A. Iqbal, and R. Shahriyar, "Banglabert: Combating embedding barrier for low-resource language understanding," *CoRR*, vol. abs/2101.00204, 2021. [Online]. Available: https://arxiv.org/abs/2101.00204

[22] A. Gasparetto, M. Marcuzzo, A. Zangari, and A. Albarelli, "A survey on text classification algorithms: From text to predictions," *Information*, vol. 13, no. 2, p. 83, Feb 2022. [Online]. Available: http://dx.doi.org/10.3390/info13020083

[23] K. Zeynalzada, "Sarcasm detection," Nov 2021. [Online]. Available: https://www.kaggle.com/datasets/theynalzada/news-headlines-for-sarcasm-detection

[24] A. Miglani, "E-mail classification nlp," Sept 2020. [Online]. Available: https://www.kaggle.com/datasets/datatattle/email-classification-nlp

[25] P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala, "Good debt or bad debt: Detecting semantic orientations in economic texts," *Journal of the Association for Information Science and Technology*, vol. 65, no. 4, pp. 782–796, 2014.

[26] J. Wang, K. Fu, and C.-T. Lu, "Sosnet: A graph convolutional network approach to fine-grained cyberbullying detection," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 1699–1708.

[27] P. K. Verma, P. Agrawal, I. Amorim, and R. Prodan, "Welfake: Word embedding over linguistic features for fake news detection," *IEEE Transactions on Computational Social Systems*, vol. 8, no. 4, pp. 881–893, 2021.

[28] D. Kotzias, M. Denil, N. de Freitas, and P. Smyth, "From group to individual labels using deep features," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 597–606. [Online]. Available: https://doi.org/10.1145/2783258.2783380

[29] H. Liang, X. Sun, Y. Sun, and Y. Gao, "Text feature extraction based on deep learning: a review," *EURASIP Journal on Wireless Communications and Networking*, vol. 2017, no. 1, p. 211, Dec 2017. [Online]. Available: https://doi.org/10.1186/s13638-017-0993-1

[30] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," 2019. [Online]. Available: https://arxiv.org/abs/1907.00503

[31] F. J. Massey Jr, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[32] C. Kwak and A. Clayton-Matthews, "Multinomial logistic regression," *Nursing research*, vol. 51, no. 6, pp. 404–410, 2002.

[33] T. Chen and C. Guestrin, "XGBoost," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2016. [Online]. Available: https://doi.org/10.1145%2F2939672.2939785

[34] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," New York, NY, USA, may 2011. [Online]. Available: https://doi.org/10.1145/1961189.1961199

[35] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[36] R. E. Schapire, *Explaining AdaBoost*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 37–52. [Online]. Available: https://doi.org/10.1007/978-3-642-41136-6_5