Fake News Detection using Machine Learning

Bongsoo Yi
UNC Chapel Hill
bongsoo@unc.edu

Chanhwa Lee
UNC Chapel Hill
chanhwa@unc.edu

Shengjie Xu
UNC Chapel Hill
shengjie@unc.edu

Wanyong Feng
UNC Chapel Hill
wanyong@cs.unc.edu

Abstract

Existing fake news detection models have been successful, but they are limited to cases where the training domain and the testing domain are similar. We tested the generalizability of various machine learning models for fake news detection by testing our models on different datasets from the training dataset. We found that different models resulted in similar prediction accuracy. The testing accuracy is high if trained on the same dataset, but not otherwise, meaning that generalizability is not decent. We then tried domain adaptation methods to build a domain-independent model, and it was successful in terms of prediction accuracy.

1 Introduction

Fake news, defined by fabricated stories with no verifiable facts, sources, or quotes, can significantly affect modern society. Several successful works have been devoted to developing fake news detection models using machine learning methods [2] [1]. However, the model that is trained on one news dataset often has poor prediction accuracy on different test datasets [11]. This may be due to the heterogeneity of the distributions of the training and the testing datasets. Therefore, we implemented both domain-dependent and domain-independent models to investigate this problem, and conducted numerous experiments on those models.

The summary of the project is as follows. First, we implemented numerous classical machine learning methods, including naive Bayes (NB), logistic regression (LR), support vector machine (SVM), and random forest (RF) to perform the fake news detection task. Second, we utilized deep learning methods such as deep neural network (DNN), one-dimensional convolutional neural network (1D CNN), bidirectional encoder representations from transformers (BERT), and long short-term memory (LSTM) to conduct the fake news detection with the same data settings as in the classical methods. Third, we analyzed the generalizability of the trained models by testing them on datasets that have different distributions from the training data (e.g., testing the models trained on longer news articles to predict the credibility of shorter, twitter styled posts). Finally, we applied transfer learning proposed by [4] to make a domain-independent model and compare the prediction accuracy with those of domain-dependent models (both types of models are trained on concatenated datasets).

2 Related Work

One particularly difficult challenge of fake news detection is domain adaptation. Specifically, the performances of models generally drop when the testing data contains different topics from training data, for example, the training data contains political news, and the testing data contains sports news; or the wording and formatting of the news is different, such as traditional media versus Twitter posts. Several studies have been dedicated to studying the possible remedies to this problem. For example, [11] proposes a framework that jointly preserves domain-specific and cross-domain knowledge of news to detect fake news from different domains; [13] proposes a domain classifier that maps the multi-modal features (text and imagery features extracted by a pre-trained BERT model) of different

Preprint. Under review.

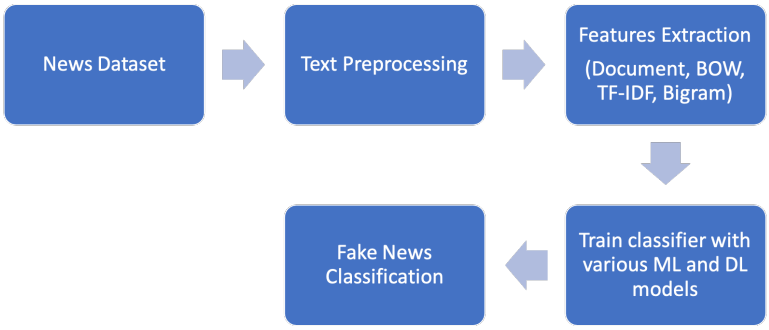

Figure 1: Overall Classification process of Fake News Detection

events to the same feature space, before feeding features in this uniformed feature space to the fake news classifier.

In the broader field of NLP, domain adaptation is also a well-studied topic, from where certain techniques may be employed to address the heterogeneity of news datasets. In general, there are three major approaches: model-centric, data-centric, and hybrid [10] Specifically, the domain adaptation method we implemented is based on [13], which develops the domain classifier parallel to the fake news classifier. The training aims to maximize the accuracy of the fake news classifier while minimizing the accuracy of the domain classifier to force the model to learn the domain-independent representations of the different datasets.

3 Data and Preprocessing

3.1 Dataset

We have used four datasets. The first dataset (later referred to as **Kaggle I**) is the Fake and Real News dataset from Kaggle. This dataset contains 21417 pieces of true politics and world news scrapped from Reuters and 23482 pieces of fake politics and tweet news scrapped from 21 Century Wire, from 2016 to 2017 [5]. The second dataset (later referred to as **Kaggle II**) consists of 5752 true articles and 4488 fake articles, which are short politics and economics statements from various websites [9]. The third dataset (later referred to as **Kaggle III**) consists of 1872 pieces of true politics news articles and 2137 fake politics news articles from various news media [7]. The fourth dataset (later referred to as **LIAR**) is a benchmark dataset for fake news detection. This dataset is split into training, validation, and test datasets, which we collapsed into one dataset. It has 4507 true short statements and 3554 fake short statements [12]. We used 60% of the **Kaggle I** dataset as training, 40% of it as testing, and the other three datasets as the testing dataset. For models using concatenated data, we combined 60% of each dataset to be a training set, and we tested models on the remaining 40 % of datasets.

3.2 Text Preprocessing

Raw news articles are preprocessed by the standard NLP preprocessing procedure. First, We removed uninformative headers or tails from the articles (e.g. **WASHINGTON (Reuters)** - (article), (article) - **Chicago Tribune**) as well as Twitter IDs, URLs, and words in square brackets. Moreover, non-alphabetic characters were replaced by spaces, and the uppercase letters were transformed into lowercase. We also removed one-letter words which might be generated by mistake. Surprisingly, we found that removing stopwords did not change the prediction result, so we remained them. The next step was using external libraries to transform words into the base form and conducting spell correction. Finally, we excluded processed articles with less than 5 words because we assumed the short article is either not informative or a source of bias.

3.3 Features

We implemented four data formatting methods that transfer the text data into a form that can be fed into the models.

Document feature is a sequence of words after preprocessing, and it is used for 1D CNN, BERT, LSTM, and domain-independent model.

Bag of Words feature is a word count vector converted from a document. Every word count vector has the same length as the size of vocabulary for a corpus of all articles, and the value of a specific element for one article indicates the number of times that word occurs in this article. Since rare words are not useful for modeling, we remained top 1000 frequent words only. To save memory, we transformed the bag-of-words matrices into sparse matrices since it contains lots of zeros.

TF-IDF (Term Frequency – Inverse Document Frequency) feature is obtained by dividing the word count vector in the bag-of-words feature by the total number of words in a document. Since it normalizes each word count vector, we could expect a more stabilized estimation of model parameters.

Bigram feature is an alternative to the Bag of Words feature, which considers two consecutive words (bigram) as one feature. The benefit of this method is that it encompasses some positional information, then it would deliver the meaning of a document more accurately. Bag of Words, TF-IDF, and Bigram features are used for RF, SVM, NB, LR, and DNN models.

4 Methods

4.1 Simple machine learning models

We implemented four simple machine learning models, namely naive Bayes, logistic regression, support vector machine, and random forest.

Naive Bayes We used the multinomial naive Bayes model with Laplace Smoothing. A probability of a word appearing in an article is parametrized and estimated by a smoothed maximum likelihood estimation, under the assumption that the appearance of two words in the same article is independent of each other.

Logistic regression Logistic regression is a linear model which is optimized by minimizing the cross entropy loss, defined by $l(\theta) = -\sum_{i=1}^{n}(y_i\phi(x_i)^T\theta + \log(\sigma(-\phi(x_i)^T\theta)))$ where n is the number of articles, y_i is the indicator of whether the i-th article is true (1) or false (0), $\sigma(t) = \frac{1}{1+\exp^{-t}}$ is the sigmoid function, $\phi(x_i)$ represents the word count vector of the i-th article.

SVM Support vector machine (SVM) is a supervised machine learning method that can be used for classification tasks. It is an algorithm that finds a hyperplane that can distinguish the data into their known labels. If the data are completely linearly separable, SVM chooses the hyperplane with the largest margin. Thus, it maximizes the closest distance from the data of each label to the hyperplane.

RF Random forest is a collection of decision trees that improves the prediction over a single decision tree. It achieves this by applying randomness into the selection of variables that will be used to construct a decision at each node. A decision tree is an acyclic directed data structure with nodes representing deterministic decisions based on variables, and edges representing paths to the next nodes based on the decision. Leaf nodes of the tree represent class labels as the output of prediction.

4.2 Deep learning methods

We implemented four deep learning models, namely Deep Neural Network, One Dimensional Convolutional Neural Network, Bidirectional Encoder Representations from Transformers, and Long Short-Term Memory models.

DNN There are various kinds of DNN models, but we focus on feed-forward neural networks, also called multi-layer perception (MLP), in this project. Using the bag-of-words, TF-IDF, and bigram features, we trained an MLP model on our training data. To apply the MLP model, we had to decide the total number of layers and the dimension of each layer. In this project, we used four layers in total. The first layer, called the input layer, has a dimension of 1000 which is the dimension of the input vector. The second and third layers, called the hidden layers, have a dimension of 64. The last

layer, called the output layer, has a dimension of 1 which represents the probability of the news is true. We also applied batch normalization and dropout layers to these layers.

1D CNN One Dimensional Convolutional Neural Network (1D CNN) is proposed by Kim [8] for sentence classification tasks using CNN models. Words in a sentence are embedded and stacked to make an embedding matrix, and convolutional filters with a width equal to the dimension of the embedding vector are applied along the rows, but not the columns. Then each feature is max pooled and be fed to a fully connected layer with dropout and finally softmax-ed to output.

BERT Bidirectional Encoder Representations from Transformers (BERT) [6] is a natural language processing model proposed by Google. It is a transformer-based machine learning model that was pre-trained from unlabeled data. BERT was pre-trained on two tasks (language modeling and next sentenced prediction) and learned contextual embeddings for the words. Using this computationally expensive pre-trained model, we fine-tuned our training dataset and optimized the performance for our specific task. Specifically, we used the pre-trained BERT model "bert-base-uncased" and fine-tuned it with Adam Optimizer for 10 epochs.

LSTM Long short-term memory (LSTM) is a recurrent neural network (RNN) architecture that is capable of retaining information over arbitrary intervals. Traditional RNNs have the "vanishing gradient" issue where the gradients of earlier inputs become close to zero as the network iterates, making it impossible to retain information over a long time. This is because the internal state of a traditional RNN passes through a non-linear function in every step of the iteration. In contrast, LSTM has a "cell state" that never passes through a non-linear function itself. Instead, only minor, linear modifications are done to the cell state through carefully controlled neural network layers called "gates". This means that the network can learn to preserve certain information by tweaking the parameters of the gates. Because of this property, LSTM is widely used in text processing, where the information provided by words early on can have an impact on words that come much later in the text.

Our LSTM model of fake news detection consists of four layers: an embedding layer, an LSTM layer, a $tanh$ non-linearity layer, and a $softmax$ classification layer. The input to the model is an N x 100 array where each article is padded or truncated to exactly 100 words. The embedding layer has a vocabulary size of 500 and embedding dimension of 100, the LSTM layer has a latent dimension of 64, and the $tanh$ non-linearity layer has 128 units. Conceptually, the embedding and the LSTM layers produce a hidden representation of the articles while the non-linearity layer and the softmax layer form a classifier.

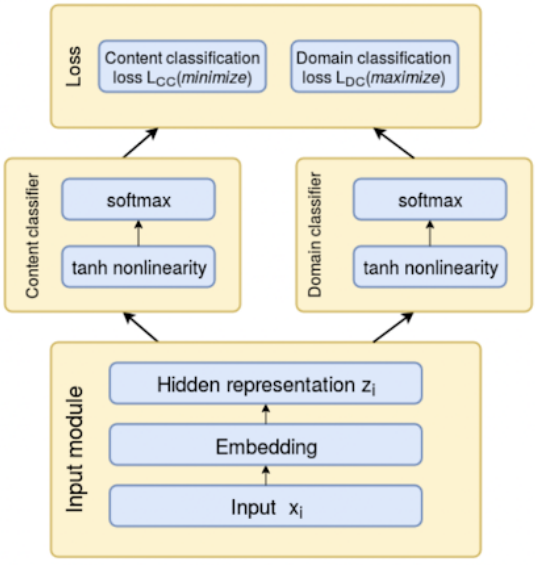

Figure 2: Structure of the domain-independent model

4.3 Domain-Independent model

We implemented the domain-independent model based on [3]. The model consists of two parts. The first part is an encoder, which consists of two layers: the first layer is an embedding layer, which embeds each word into an array of size 100; the second layer is an LSTM layer which encodes the embedding data into the size of 128. The second part consist of two densely connected layers; one for content classification, and another for domain classification. The loss function is the combination of content classification and domain classification loss. The goal of training is to minimize the classification loss while maximizing the domain classification loss because we want the model to make accurate predictions concerning the credibility of the news and does not learn domain-specific features.

5 Results

5.1 Non-domain adaptation models

We trained the traditional machine learning models (RF, SVM, NB, LR) and deep learning models (DNN, 1D CNN, BERT, LSTM) on the Kaggle I training data. In Figure 3, the plot represents the testing accuracy of 1D CNN, BERT, and LSTM models trained using document features on each testing domain (Kaggle I, Kaggle II, Kaggle III, LIAR). The testing accuracy is defined by the proportion of the data correctly classified in all testing data. Overall, we can see that the performance on Kaggle I was significantly better than the other testing domains. This is quite obvious since we used the training data of Kaggle I to construct the models. Also, it showed decent performance on Kaggle III while it showed an accuracy of around 50% on Kaggle II and LIAR data. One takeaway here is that among the three models in Figure 3, the higher accuracy on Kaggle I, the lower accuracy on Kaggle II and LIAR data. Thus, there isn't an outstanding model that overwhelms the other two models' performance.

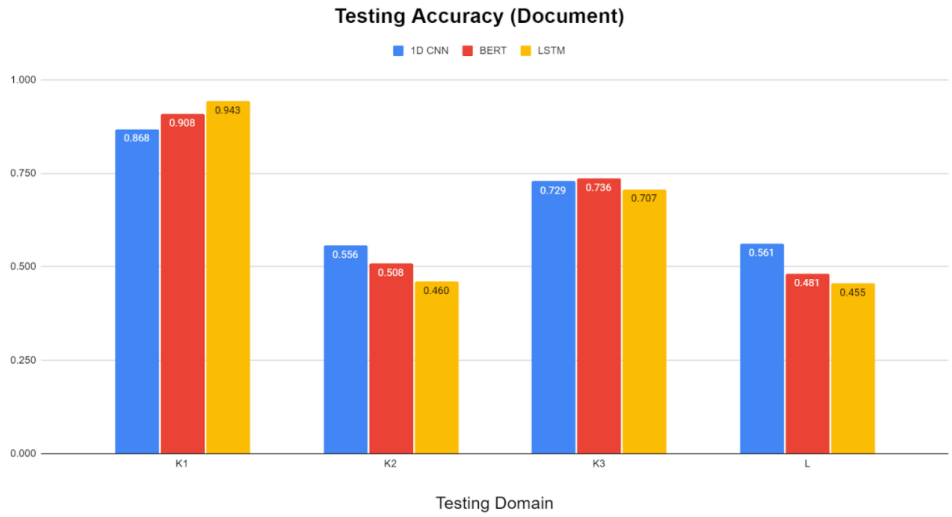

Figure 3: Test accuracy trained on dataset Kaggle 1 (Document feature)

With the BOW feature, we trained RF, SVM, NB, LR, and DNN on Kaggle I training data and compared their test accuracy. Figure 4 shows the test accuracy trained with the BOW feature. The results generally showed a similar tendency with the document feature. We obtained superior accuracy of nearly 100 percent on Kaggle I and comparably good performance on Kaggle III. In contrast, it showed lower accuracy than 50% on Kaggle II and LIAR. Interestingly, we observed that all five models have similar accuracy on each testing data. To dive deep into the data, we also trained the models on Kaggle II, Kaggle III, and LIAR with the SVM model. Figure 7 shows the test accuracy of SVM trained on Kaggle II, Kaggle III, and LIAR, respectively. We recognize that the test accuracy distributions trained on Kaggle I and III are similar and that Kaggle II and LIAR are also similar.

Intuitively, we can think that Kaggle I and III have similar types of data, and Kaggle II and LIAR as well.

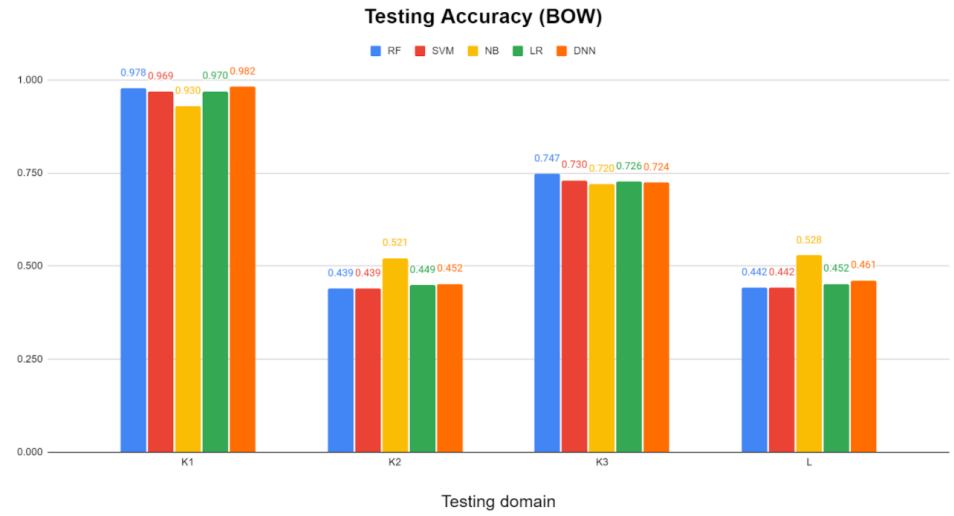

Figure 4: Test accuracy trained on dataset Kaggle 1 (BOW feature)

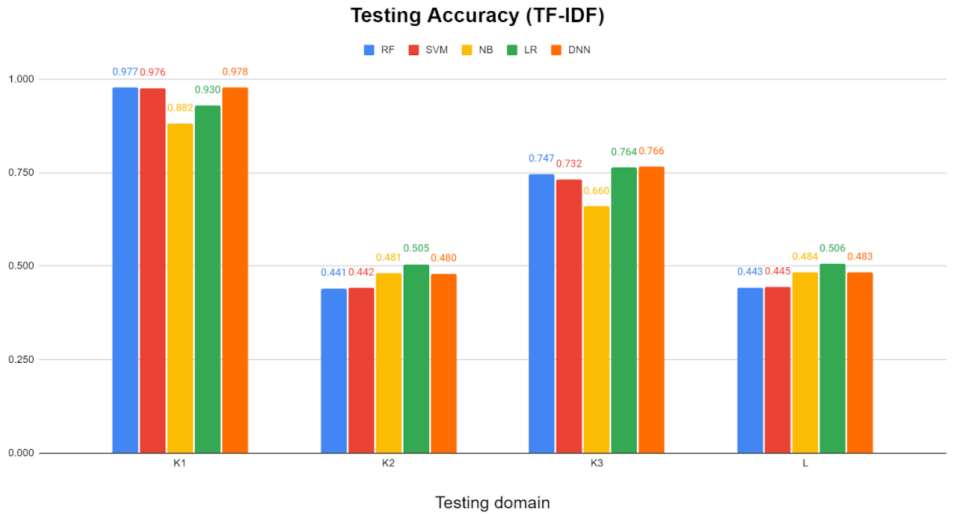

Figure 5: Test accuracy trained on dataset Kaggle 1 (TF-IDF feature)

The analysis with the TF-IDF and bigram features showed similar results with the analysis with the BOW feature: 1) showing relatively good results on Kaggle I and Kaggle III, and poor performance on Kaggle II and LIAR, 2) all five methods show similar test accuracy on each testing dataset.

5.2 Domain adaptation model

We trained both domain-dependent and domain-independent models on the concatenated dataset of Kaggle I, II, III, and LIAR, and then tested these models on the concatenated dataset as well as on Kaggle I, II, III, and LIAR individually.

In Figure 8, the plot represents the testing accuracy of RF, SVM, NB, LR, LSTM, and the domain-independent model. To compare the results, we first calculated the average accuracy of the six methods on each testing domain. We observed that the combined dataset had the second-best average accuracy among all the testing domains. It is reasonable because the majority of the data

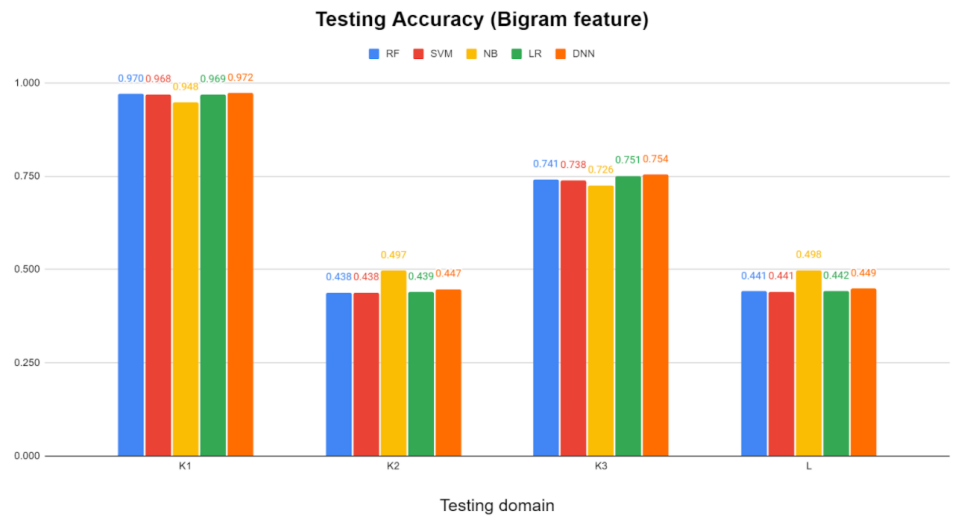

Figure 6: Test accuracy trained on dataset Kaggle 1 (Bigram feature)

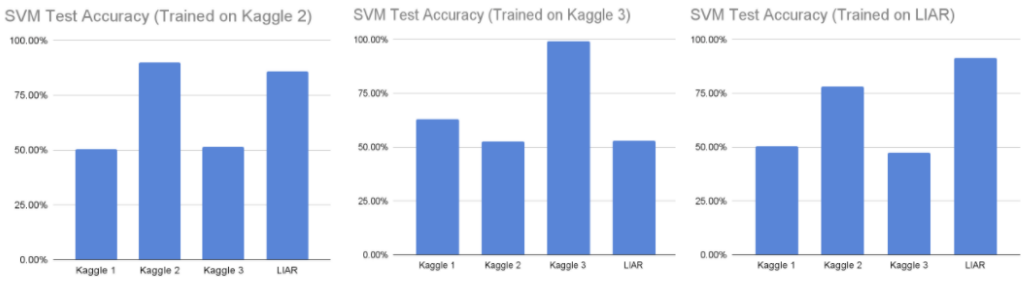

Figure 7: Test accuracy trained on dataset Kaggle 2, Kaggle 3, and LIAR with SVM model (BOW feature)

comes from the Kaggle 1 dataset, on which all the models have the best performance. Moreover, the performances of all the methods increased compared to the previous result, because the training data comprises the portion of all the datasets. We also found that the domain-independent model beats nearly all the domain-dependent models, which implies that the model learned the independent representations of different domains. However, the surprising result was that RF performed better than the domain-independent model on all the datasets, which requires further investigation to explain.

6 Discussion and Conclusion

We focus on three aspects: characteristics of different datasets, the results from different methods and features, and domain adaptation.

Firstly, the datasets have some interesting variations among them. Both Kaggle 1 and Kaggle 3 consist of fairly long articles. We suspect that this is the main reason why they achieved higher accuracy when trained and tested on the same domain shown in Figure 7. In contrast, Kaggle 2 and LIAR consist mostly of shorter, typically one-line sentences. This means that there are not as many characteristics that can be derived from the text alone. Since we did not have external fact-checking and relied solely on the patterns in the text itself, we ended up with worse accuracy for these two datasets.

Furthermore, we were surprised that there are only minimal differences among the different models. Particularly, the simpler models like LR achieved similar accuracy as the much more complex models

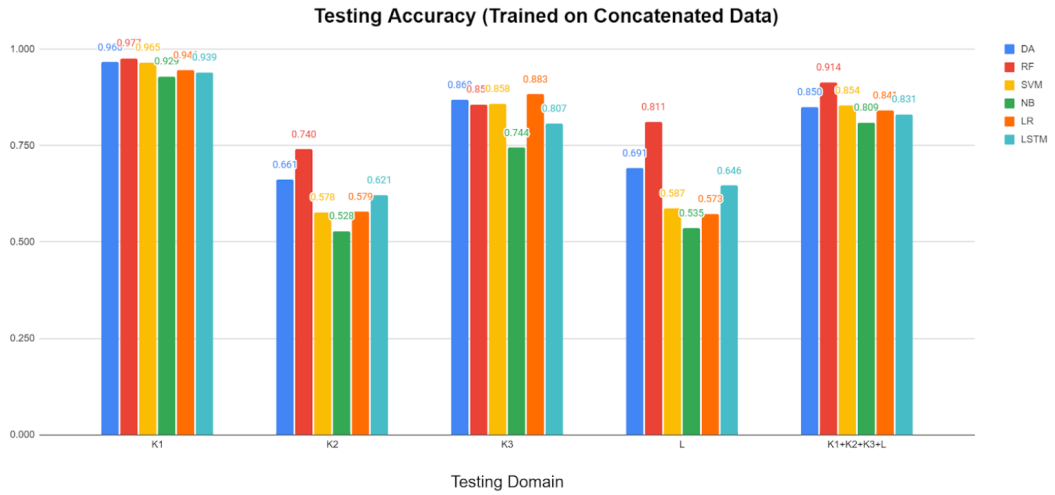

Figure 8: Test accuracy trained on the concatenated dataset

such as BERT. This hints that the bottleneck in our experiments is not the complexity of the models since even the simple models are reaching about 95% accuracy. Similarly, using different features, such as document features compared to BOW features, didn't result in a big change in accuracy either. These results all point to the possibility that the domain difference is the dominant factor here for the poor cross-domain accuracy. This leads to our third point about domain adaptation.

Finally, we learned that in tasks like this, domain adaptation plays a big role in designing an effective machine learning model. Simply concatenating all the domains and using it as the training set yielded a clear increase in accuracy, even it does not consider domain-specific information. A domain-independent model that tries to eliminate domain-specific information further improved the performance, albeit only by a small margin. We now appreciate the importance of domain adaptation much more than before we started the project.

In the future, we wish to implement a true domain adaptation method that does not use the labels for the target domain. We think this will be more valuable in the real world, as we believe there are many cases where we have data in a specific domain but not the associated labels, and we want to be able to train a classifier that will perform decently on that domain.

References

[1] Muhammad Ahmad Iftikhar Yousaf. "Fake News Detection Using Machine Learning Ensemble Methods". In: *Probab. Surv* 2.245 (2005), pp. 10–1214.

[2] Supanya Aphiwongsophon and Prabhas Chongstitvatana. "Detecting Fake News with Machine Learning Method". In: *2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. 2018, pp. 528–531. DOI: 10.1109/ECTICon.2018.8620051.

[3] arnavc1712. *Cross-Domain-Fake-News-Detection*. https://github.com/arnavc1712/Cross-Domain-Fake-News-Detection. 2019.

[4] Hosein Azarbonyad, Robert Sim, and Ryen W. White. "Domain Adaptation for Commitment Detection in Email". In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. WSDM '19. Melbourne VIC, Australia: Association for Computing Machinery, 2019, pp. 672–680. ISBN: 9781450359405. DOI: 10.1145/3289600.3290984. URL: https://doi.org/10.1145/3289600.3290984.

[5] Clement Bisaillon. *Fake and Real news dataset*. URL: https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset.

[6] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[7] jruvika. *Fake News detection*. URL: https://www.kaggle.com/jruvika/fake-news-detection?select=data.csv.

[8] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *Association for Computational Linguistics* Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014). URL: https://aclanthology.org/D14-1181.

[9] Saivenket Patro. *Fake News Detection Dataset*. URL: https://www.kaggle.com/ksaivenketpatro/fake-news-detection-dataset.

[10] Alan Ramponi and Barbara Plank. "Neural Unsupervised Domain Adaptation in NLP - A Survey". In: *CoRR* abs/2006.00632 (2020). arXiv: 2006.00632. URL: https://arxiv.org/abs/2006.00632.

[11] Amila Silva et al. "Embracing Domain Differences in Fake News: Cross-domain Fake News Detection using Multi-modal Data". In: *CoRR* abs/2102.06314 (2021). arXiv: 2102.06314. URL: https://arxiv.org/abs/2102.06314.

[12] William Yang Wang. *LIAR: A Benchmark Dataset For Fake News Detection*. URL: https://github.com/thiagorainmaker77/liar_dataset.

[13] Tong Zhang et al. "BDANN: BERT-Based Domain Adaptation Neural Network for Multi-Modal Fake News Detection". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9206973.