HYPERPARTISAN NEWS DETECTION

ROBIN LIEBERT, University of Stavanger, Norway
KATHIR TAHASIN, University of Stavanger, Norway
ARMIN SABRI, University of Stavanger, Norway
ELISABETH EIK, University of Stavanger, Norway

As information access has become more readily available from a larger variety of sources over the last few decades, a poignant trend has become more intensely apparent: objective information authority loses it's external consensus of command as the veracity of the source loses imperative authority. Herein lies the predicament that is increasingly being faced by news sources, a traditionally routine source of implicitly reliable information: hyperpartisan news. As the number of channels of news information access are ever increasing and the polarity and leanings of said sources become increasingly more intense, a critical question arises: What is valid information, and by extension, what is and is not therefore 'unreliable' news, or even fake news? Hyperpartisan news, which leans far to heavily on the left or right side of the political spectrum, is therefore of great interest. BuzzFeed conducted a research effort to manually collate a repository of such hyperpartisan news sites, and we conducted data mining research to validate procedurally their work, as well as test Machine Learning models unto this data set to emulate and further predict future sites, and investigate the possibilities of such methods in this field.

1 INTRODUCTION

Hyperpartisan news is defined as extremely biased in favor of a political party, either right or left winged. [Definition.org 2022] Hyperpartisan news is often labeled, and accused, of the crime of falling under the nebulous banner of 'fake news'. Increasingly however, academic circles have come to acknowledge the problems and hurdles presented at large by the intense stream of misinformation and in-accurate information being presented as objective, thorough facts.

To combat this, many efforts have been undertaken, both outside and on either side of the political spectrum, to address this issue. Faktisk.no is a norwegian web page which does fact-checking in Norwegian, for example, for the Norwegian news-consumer base. This is because misinformation and 'fake news' have major, documentable influence and real-world ramifications on consequential issues such as major political elections and discourses. [Faktisk.no 2022]

Authors' addresses: Robin Liebert, University of Stavanger, Kjell Arholms gate 41, Stavanger, Norway, r.liebert@stud.uis.no; Kathir Tahasin, University of Stavanger, Kjell Arholms gate 41, Stavanger, Norway, k.tahasin@stud.uis.no; Armin Sabri, University of Stavanger, Kjell Arholms gate 41, Stavanger, Norway, a.sabri@stud.uis.no; Elisabeth Eik, University of Stavanger, Kjell Arholms gate 41, Stavanger, Norway, el.eik@stud.uis.no.

0730-0301/2022/5-ART111 $15.00

In this project we shall use the resources provided by one such research project, BuzzFeeds' hyperpartisan news site labeling effort, to experiment with different machine learning techniques such as neural networks, adaboost, and LSTM, and measure their performance. We shall also investigate the veracity and efficacy though LSH of the data curation effort of BuzzFeed's research team themselves, who wholly committed to a data curation effort and not any degree of machine learning work. We go further through the data set in chapter 4. Ultimately therefore, hyperpartisan news detection entails the task of predicting whether the news is right winged or left winged, and all the ML and data cleaning/curation tasks associated thereof.

Our implementation and combined-work setup can be found in this git repository: Group05 Repository.

2 TASK DEFINITION

The task is defined as follows:

Use the labelled data for hyperpartisan news detection from Buzzfeed's provided Github repository and train a classifier to detect whether the website is right or left biased. We use three data set found in Github containing information about the websites, information whether the website is right or left and information about the Facebook pages' statuses. We shall also in parallel investigate the accuracy of the manual labeling through data algorithmic bucketing techniques.

In this project we are going to try to classify whether the web pages are right or left winged, based on different machine learning techniques. Neural network, adaptive boosting and support vector machine are some of the techniques that will be discussed later in the report.

3 CONTRIBUTIONS

Everyone in this project contributed equally to the project. The section was mostly divided like this

- **Pre-processing** Pre-processing of the metadata was performed by Elisabeth. Armin did the word2vec, Robin did the universal sentence encoder and Kathir did LSH investigation.
- **Classifiers** Robin, Armin and Kathir worked mostly on the different classifiers. Elisabeth helped with the results and discussion.
- **Visualization** Everyone contributed to the visualization.

The rest of the report is organized as follows: Chapter 4 contains information about the data set and chapter 5 contains the different pre-processing techniques that were used on the data set. Then the next section is about the theory behind the different classification techniques, which is chapter 6. Chapter 7 describes our results and the different approaches that were tried, and the discussion of these

methods can be found in chapter 8. Finally the last chapters are chapter 9 conclusion and chapter 10 further work.

4 DATA

4.1 Data set

The data of interest was produced by a BuzzFeed research project, with all associated elements found on a provided GitHub repository, and is split in three different data sets of primary information, which is domain, page_info and sites. There is also a separate set of files (folder), containing a separate CSV file of all Facebook updates for each site's corresponding Facebook page, for a period of time, for each site. These contain the bulk of raw data for ML applications.

The first primary data set, domain, contains information about the web pages, its contact information (admin, registrant and technical) and additional emails. The second primary data set, page_info contains information about the Facebook pages corresponding to the websites. It has information such as information about the Facebook page, fan count and page id. The third and last primary data set is sites. This data set contains the websites, whether the website is right or left winged, Facebook id, if the website is Macedonian and an column called unavailable_id.

4.2 Investigating Data Set Creation Veracity using LSH

As per the source article and repository whence this research is based, we are informed that the 677 individual websites that are listed were all manually labeled with partisan leanings. Buzzfeed researchers manually crawled through these sites and human-judgment deduced their leanings summarily from initial gleanings. We were also provided with a large set of CSV files that were enumerated with page ID's, to indicate a large text-set of facebook updates on the facebook pages associated with these websites. Thus we are presented with two relevant datasets: a set of websites labeled for their partisan leanings, and a set of csv files, each containing a large number of facebook updates for each site.

It is therefore imperative the sites have facebook_id so as to be able to match the site to its appropriate CSV files containing the main data used for training and other associated purposes. However, a large number of them were missing this and other important properties. Therefore only 452 usable sites were available for further analysis. However, due to the manual nature of the labeling, some preliminary analysis was done using locality sensitive hashing (LSH) to review the veracity of this labeling process during pre-processing.

5 PRE-PROCESSING

It was necessary to perform pre-processing to the data set before we used it in the different data models. The data set contained a lot of NaN values, where some columns had over 90% missing values, and some rows had most of the columns filled with NaN values.

In the next sub chapters we are going deeper in the different pre-processing techniques that were done on the data sets.

5.1 Data cleaning

We started the pre-processing of the dataset by cleaning it from NaN values. This was done by going through the three different data sets. To begin we removed all columns which had more than 80% NaN values, and the same were done for the rows where the threshold for missing values were sat to 50%, i.e., all rows having more than 50% missing values were removed.

In the next step we looked for patterns in the data set to see if we could fill some missing NaN values with either the mean or mode. Some were also filled with 0, as Postal code if there were none or phone number if there were no matching we could fill.

There were some columns that were close duplicates of each other, like column '3' and column 'domain'. Column '3' contained 'https://whois.domaintools.com/' before the webpage, and domain contained only the webpage. For the webpage column in page_info we removed the http:
www to make it easier to merge the data sets.

After pre-processing and cleaning the data set, we merge them all together based on the web pages. The merged data set that was used only contained 1/3 of the original data set because of mismatch with the web pages when combining them. This was en effect after merging with the sites data set, which contained information whether the web page was right or left winged, and is an important attribute to the classifiers.

After the pre-processing we saw that by looking at the macedonian column that contains information whether the site has it's origin in Macedonia or not, it is a 97.5% chance that the site is right winged if it has it's origin there. This could be useful information in the classifiers when classifying the websites.

5.2 Data Set LSH investigation

Taking inspiration from previous assignments, LSH was used to similarity bucket the websites based on their texts. Each site was treated as a single 'article' by compressing all statuses into a single one-line feature, and then tokenizing that one-line feature for LSH analysis and filtering through Jaccard similarity.

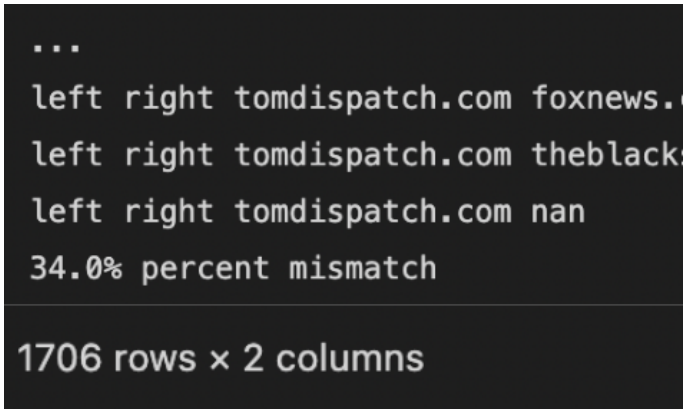

Fig. 1. N=1, MC=9000, T=0.4; LSH paramater investigation.

Three key parameters were tweaked in this process. Ngram length (N), x number of top ngrams (MC), and Jaccard similarity threshold (T). These were tweaked to see the relationship between phrases and words in the similarity relationship between the sites. The goal here is to analyze and understand the relationship between articles

through their similarity under different parameters, and through this investigate the nature of the partisan sites further, before we commence further unsupervised machine learning paradigms to produce predictive results. The aim therefore is to deconstruct whether the labeling was indeed done correctly, and if so, what further dimensions might the sites fall under past the binary left/right dichotomy they are corralled into.

5.3 Text preprocessing

Some machine learning techniques are not able to use categorical data, and need to be converted to numerical values. To do that we have used two different methods, which is explained in the subsections below.

5.3.1 Word2Vec.

Word2Vec is a Natural Language Processing technique proposed by Google in 2013 [Tomas Mikolov 2013]. The Word2Vec process the text data by producing word embedding using a neural network model. The main principle of this method is to captures both syntactic and semantic similarities between the words. Once feeding the large text corpus into one learning model, the algorithm converts it to a set of high dimensional vectors by calculating the cosine distance between words [Naili et al. 2017].

The algorithm mainly consists of two learning models, Continuous Bag of Words (CBOW) and Skip-gram. CBOW predicts target words based on their context, while Skip-gram predicts target context based on words [Ma and Zhang 2015]. The difference between the models can be seen in fig. 2 [Tomas Mikolov 2013]. In this project, an open-source python library called Gensim has been used to develop word embedding by training Word2Vec with either CBOW or Skip-gram model.

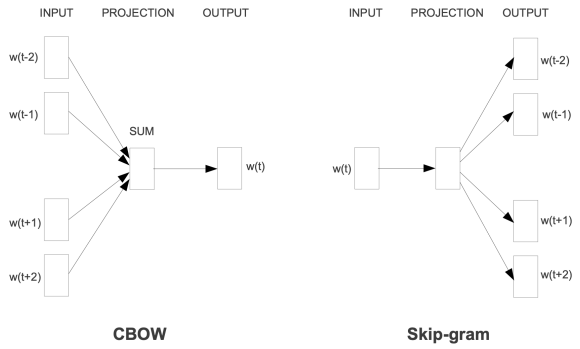

Fig. 2. Word2Vec main learning model [Tomas Mikolov 2013]

5.3.2 Universal Sentence Encoder - Semantic Similarity.

USE-4 is a model which encode textual data into high dimensional vectors based on Semantic Similarity. This is practical in this task compared to using n-grams, or similar algorithms, due to the semantic similarity analysis text which is similar would be close to each other in vector space. USE-4 returns a 512 dimensional vector for encoded text, to reduce complexity and still keep the relations between different pages the dimensions have been reduced to two dimensions. In this project t-distributed stochastic neighbor embedding (t-SNE) has been used for dimensionality reduction as USE-4 already vectorized by similarity.

By normalizing the data points and perform histogram stretching to use the full range of 0-255, the numerical data can be reshaped and plotted as an image. In Fig. 3 the mean images of left and right wing sites can be seen after being Gaussian filtered.

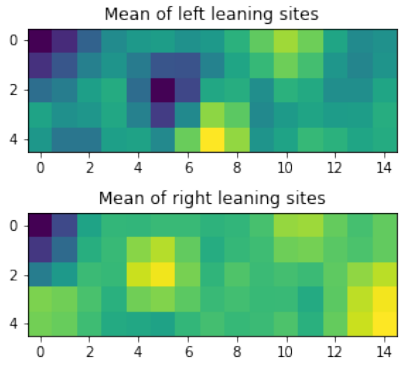

Fig. 3. Gaussian filtered mean images.

5.3.3 Discrete Fourier Transform.

Discrete Fourier Transform is a algorithm for converting data from it's original domain to the frequency domain. In the frequency domain the under laying frequencies in the data and their magnitude can be analyzed. In this project NumPy's built in function FFT2 has been used. By analyzing the phase of the two classes in Fig. 4 and Fig. 5 it's clear two see that the fundamentals of the classes are widely different.

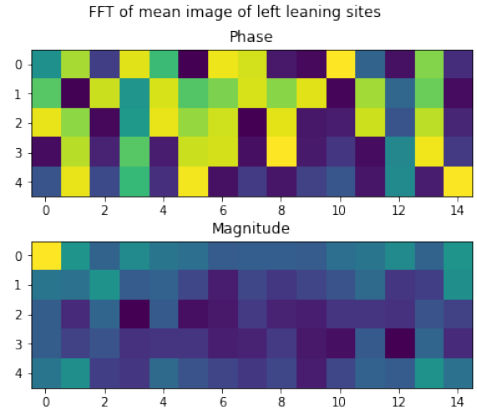

Fig. 4. Discrete Fourier Transform of left leaning sites.

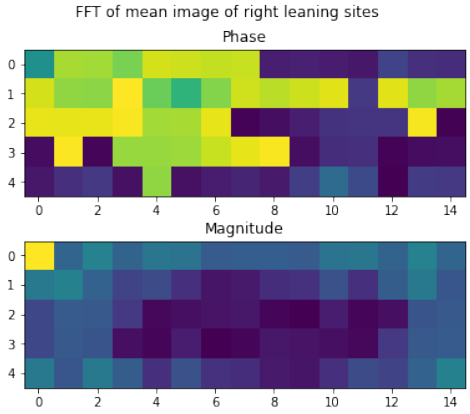

Fig. 5. Discrete Fourier Transform of right leaning sites.

With the difference in phase validating the disparity of the classes in the frequency domain, the process of building a classifier utilizing CNN is corroborated.

6 THEORY

This chapter is about the theory behind the classifiers and algorithms that have been used in this project, which is divided in one subsection for each type of classifier and algorithm.

6.1 Jaccard Similarity

During the initial investigative section using Locality Sensitive Hashing, a key metric was identifying similarity through the Jaccard Similarity scoring methodology. Identifying pairs of similar instances is also called similarity joins, something relevant to our processing. How this is done is that with a set of data instances, a similarity threshold J, and a join attribute a, the goal of similarity joins is to find all pairs of instances A, B, where their similarity on the join attribute is larger than the threshold J (i.e., sim(A.a, B.b) J). There are various similarity measurements, but in this work we focus on Jaccard similarity, which is proven to be successful for high dimensional, sparse feature sets. Jaccard similarity for two feature vectors S1 and S2 is defined as

$$sim(S1, S2) = \frac{|S1 \cap S2|}{|S1 \cup S2|}.$$

[Gupta 2018]

6.2 Neural network

Neural network (NN) is a machine learning technique and works similarly to the human brain's neural network. It consists of input layer, a number of hidden layers and an output layer. The number of hidden layers depends on the network, and it can be both deep with many layers, which is called deep neural network, or wide with many nodes in each layer, called wide neural network. The input layer collects the input patterns, and the output layer has classifications or output signals to which the input patterns may map. The hidden layers are fine-tuning the input weightings until the network's margin or error is minimal. [Chen 2018]

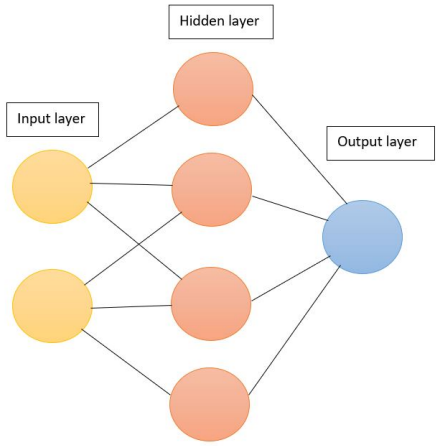

Fig. 6. Example of neural network with 1 hidden layer

6.3 Convolutional Neural Network

Convolutional Neural Network (CNN) is a deep learning method mostly used in computer vision systems. It's built up of a number of filter kernels filtering the input image for traits and creates new dimensions based on each filter. It's common to include layers of Max-Pooling and Padding between the layers of convolution to keep the main traits and keeping a minimum size on the input to the convolution layer. To perform classification it's necessary to flatten the data and use a common dense layer to reduce the size before having a single neuron performing the binary classification if that is the use case. [Saha 2018]

6.4 Long short-term memory

Long short term-memory, also called LSTM, is a type of recurrent neural network in deep learning. It is capable of learning order dependence in sequence prediction problems, and can learn to keep only relevant information to make predictions, i.e. forgets non relevant data.

The operations happening inside the LSTM's cells are different from a recurrent neural network as the operations is allowing the network to keep or forget information. The cell state acts as a transport that transfers relative information down in the sequence chain. As the cell state goes through the sequence, some information gets added and some get removed to the cell state via gates, where the gates are different neural network. The gates decide which information is allowed on each cell state, and learn what information is relevant or not during training.

Inside the gates there are sigmoid activation functions, which is similar to the tanh activation function which is found in the reccurant neural network. Compared to the tanh function which get values between -1 and 1, the sigmoid will get values between 0 and 1. This is helpful to update or forget data, because multiplying any number with 0 results in 0, causing values to disappear or be "forgotten", and multiplying with 1 results in 1, and the value is "kept". The network then learns which data is important, and not.

LSTM contains different kinds of gates, which is forget gate, input gate and output gate. The forget gate decides what information is relevant from the prior steps, input gate decides the relevant information to be added, and output gate determines what the next hidden state should be. [Phi 2018]

6.5 Adaptive boosting

Adaptive boosting is another classifier, also called AdaBoost. It is most commonly used with decision tree with one level, i.e. only one split, which is also called decision stumps. [Saini 2021]

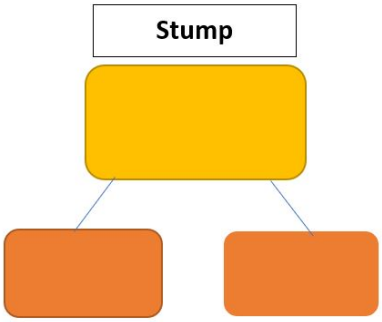

Fig. 7. Decision stump

The algorithm builds a model and gives equal weight to all data points. Misclassified items are assigned higher weights, so that it is given more importance in the next model. [Saini 2021]

After each classifier is trained, the weight is assigned to the classifier as well based on accuracy. The more accurate classifier is assigned higher weight so that it will have more impact in the final outcome. A classifier with 50% accuracy is given a weight of 0, and a classifier with less than 50% accuracy is given negative weight [Desarda 2019]

To use AdaBoost we split the data set into test and train set by using scikit learn's train_test_split, which takes test_size and train_size as inputs. The split up data set is used by Scikit learn's Adaboost algorithm, which takes n_estimators (number of trees) and learning_rate as input [Learn 2022]

6.6 Support vector machine

The main idea of support vector machine (SVM) is to find a hyperplane in an N-dimensional space, that distinctly classifies the data points. There are many possible hyperplanes (decision boundaries) that could be chosen to separate the data points, and the goal is to find the plane that has the maximum distance between the data points of both classes. The data points falling on either side of the hyperplane can be attributed to different classes.

Support vectors are data points that are closer to the hyperplane. These influence the position of the hyperplane and the margin of the classifier will be maximized. If the support vectors are deleted it will change the position of the hyperplane. These data points help build the SVM.

The support vector machine takes the output of the linear function and classifies the output to 1 or -1, depending if the output is larger or smaller than 1 or -1, respectively.

[Gandhi 2018]

6.7 Local sensitive hashing

Local sensitive hashing, also called LSH, which is a family of functions that hash data points into buckets so the data points near each other are located in the same buckets with high probability. LSH can be broken down to 3 broad steps:

- **Shingling**
- **Min hashing**
- **Locality-sensitive hashing**

Shingling convert each document into a set of characters of length k (which is also known as k-shingles), where the idea is to represent each document as a set of k-shingles. Similar documents are then more likely to share more shingles. Jaccard index is often used at a measure of similarity between documents, where more number of common shingles return a higher Jaccard index, and hence the documents are similar.

Hashing is then used to convert each document to a small signature using a hashing function. When using Jaccard similarity the appropriate hashing function to use is min-hashing. Min-hashing hashes columns of signature matrix M by using seceral hash functions. If two documents hashes into the same bucket for at least one of the has function. these two documents become a candidate pair. By using min-hashing it saves space, and at the same time preserving the similarity.

LSH tries to find algorithms such that if we input the signature of 2 documents, it will tell wheter or not they form a candidate pair, i.e. if their similarity is greater than a certain threshold. [Gupta 2018]

7 RESULTS

7.1 LSH investigation

7.1.1 Top Ngrams factor.

Preliminary statistics can be broken down into parameter sensitivity across three categories, namely the three parameters in question. For the top ngrams parameter, there is a positive correlation with the number of top most common ngrams investigated and accuracy. The most common ngrams to be found is called the MC value. Accuracy in turn has a negative correlation with the number of matches returned for each threshold level (i.e. all other parameters being equal). It is thus one of the less significant findings, but of clear point: piling through a higher number of common ngrams result in fewer articles that are found to be similar, but those that are found are more thoroughly similar. To illustrate, at ngram length of 3 (N value) and Jaccard similarity threshold of 0.4 (T value), parsing the top 500 most common (MC) ngrams results in 140 matches at 11% mismatch (matches where the left/right polarity does not match); MC value of 1000 result in 70 matches at 4% mismatch rate, and 5000 MC results in 18 matches with no mismatch at all.

...	...
left right msnbc.com ne	right right freedomfor
left right msnbc.com na	right right freedomfor
left left counterpunch.	left right msnbc.com w
11% percent mismatch	4% percent mismatch
140 rows × 2 columns	70 rows × 2 columns

Fig. 8. N=3, T=0.4; MC=500 vs MC=1000 comparison.

7.1.2 *Jaccard Similarity Factor.*

Jaccard threshold naturally has the same positive correlation to accuracy, as this is the most direct metric of filtering through accuracy. At N value of 1 and MC value of 9000, T value (threshold) of 0.4 initially returns 1706 separate match instances at 34% mismatch, and upping it to 0.9 T returns 362 matches at 24% mismatch. This also has less significance in terms of text characteristics as it just acts as a direct filter to increase accuracy. It was largely expected and provides little insight.

7.1.3 *Ngram Length factor.*

The final and more significant parameter findings were within the ngram lengths N. As the N value is pushed up, from 1 to 3, the accuracy goes up in the same positively correlated manner. For the same MC and T values, as N value goes up with fewer and fewer matches. Mismatch rates, however, do not scale as linearly, with the jump down in mismatch rates from N values of 1 to 2 is less than the mismatch rate decrease from 2 to 3.

7.2 Webpage based classification

7.2.1 *Classification using Support Vector Machine (SVM).*

By auto selection of the gamma parameter, SVM gave an accuracy of 0.7242.

7.2.2 *Classification using AdaBoost.*

Using SigOpt for optimizing the optimal learning rate: 0.04296 and number of estimators: 94. This gives a accuracy of 0.734783. The trends during tuning can be seen in Fig. 9 (yellow line indicate optimal parameters).

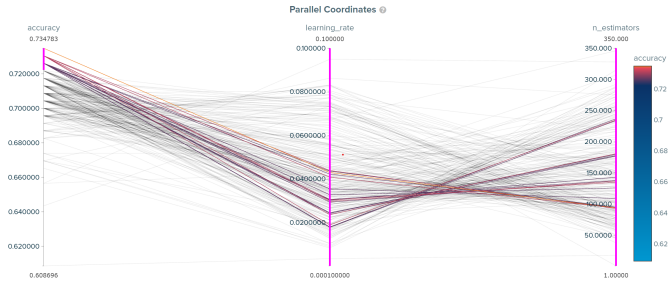

Fig. 9. 250 trials during tuning of AdaBoost

7.2.3 *Classification using Artificial Neural Network (NN).*

It was two different NN designs used for this classification, with two hidden layers and no hidden layers. The hyperparamters of the one with the hidden layers was:

- Input layer: 50 units, Activation: sigmoid
- 1. hidden layer: 40 units, Activation: relu
- 2. hidden layer: 10 units, Activation: relu
- Output layer: 1 unit, Activation: sigmoid

This gave optimal results after one epoch of training with learning rate of 0.01. The results were:

Loss: 0.5990, Accuracy: 0.7241

Without any hidden layer the optimal design is:

- Input layer: 288 units, Activation: relu
- Output layer: 1 unit, Activation: Sigmoid

This gave optimal results after five epochs of training with learning rate of 0.001. The results were:

Loss: 0.5673, Accuracy: 0.7414

7.2.4 *Classification using Convolutional Neural Network (CNN).*

The classification using CNN was done in similar fashion as with NN. One model had several convolutional layers with Max-Pooling and padding between, while the other was just one layer of convolution. Both models has a Dense layer with 12 units before the single unit output layer.

Result of complex model:

Loss: 0.5894, Accuracy: 0.7241

The result of the simple model was very similar:

Loss: 0.5760, Accuracy: 0.7241

7.3 Facebook status based classification

7.3.1 *Classification using Support Vector Machine (SVM).*

Small data set of 10 000 statuses using USE-4. By auto selection of the gamma parameter SVM gave an accuracy of 0.79.

7.3.2 *Classification using AdaBoost.*

The data set of 10 000 statuses the AdaBoost parameters were optimized using SigOpt.The optimal learning rate found: 0.030176 with 350 estimators, the accuracy is 0.793974. But tens of other combinations managed the same accuracy (the yellow line indicate optimal parameters), as can be seen in Fig. 10

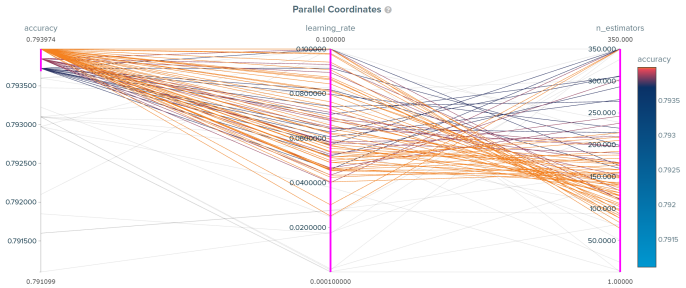

Fig. 10. 90 trials during tuning of AdaBoost

7.3.3 Classification using Long Short Term Memory(LSTM). Original data set of 4 000 000 statuses using CBOW model of Word2Vec technique. The hyperparamteres for the Neural Network with two hidden layers was following:

- Input layer: Embedding layer with an integer matrix of vocabulary size as input.
- 1.hidden layer : LSTM with 128 units, Activation: tanh
- 2. hidden layer: Dense with 32 units, Activation: relu
- Output layer: Dense 1 unit, Activation: sigmoid

Our model achieved an accuracy of 0.8136 after 4 epochs of training. The results were:

Loss: 0.4362, Accuracy: 0.8136

7.3.4 Classification using Neural Network. Small data set of 10 000 statuses using USE-4. The hyperparamters for the Neural Network with four hidden layers was:

- Input layer: 30 units, Activation: relu
- 1. hidden layer: 60 units, Activation: sigmoid
- 2. hidden layer: 60 units, Activation: relu
- 3. hidden layer: 20 units, Activation: tanh
- 4. hidden layer: 70 units, Activation: tanh
- Output layer: 1 unit, Activation: Sigmoid

This gave optimal results after two epochs of training with learning rate of 0.001. The results were:

Loss: 0.4795, Accuracy: 0.7925

The same neural network architecture was also used for data set with 4000000 statuses using keras.tokenizer . The optimal results after 5 epochs with learning rate of 0.001 were following:

Loss: 0.5062, Accuracy: 0.7955

7.3.5 Classification using Convolutional Neural Network (CNN). Using the data set with 10 000 statuses with full 512 features from USE-4 gave the following result:

Loss: 0.5361, Accuracy: 0.7855

In Fig. 11 the architecture can be seen with just a single convolution layer before it's flatten. Several attempts with increased complexity did not increase test accuracy.

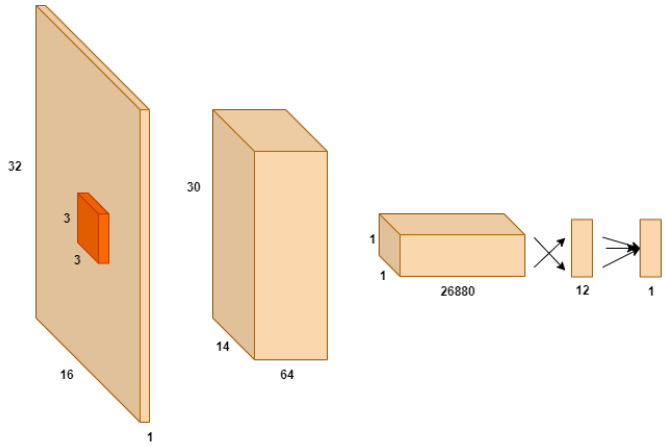

Fig. 11. Layers: 3x3 Conv2D layer with relu activation, flatten layer, fully connected 12 units with relu activatin, output layer with sigmoid activation

8 ANALYSIS

To design and tune a model for hyperpartisan detection is a complex task. Without any ground-truth or best practice the different options of solving this task is limitless. As can be seen in chapter 4 Data, it was used several data sets to solve the classification. There was two types of data sets that the works was performed on, metadata of the web pages and the scraped data from the Facebook pages. To discuss the decisions made this chapter are split into several parts: pre-processing, data-set source analysis, choice of models, tuning the model.

8.1 Pre-processing

After cleaning the data set it was necessary to convert the categorical data into numeric data. There are several techniques covered in this course to perform this conversion, and the most prominent ones are one-hot encoder and word2vec (w2v). Due to the large spread of categorical data the group found w2v to be most suited for the task. To get a reference a second model based encoder, USE-4, was chosen for it's semantic similarity properties. To reduce the complexity in the 512-D vector created from USE-4, it was reduced to 2-D with t-distributed stochastic neighbour(t-SNE). When reducing the dimensionality the algorithms available is hard to differentiate between. In this case we ended up with comparing Principal Component Analysis (PCA) as they are both unsupervised methods for dimensionality reduction. The choice of using t-SNE was made because of it's capability of keeping the relative distance between the datapoints and how it handles outliers.

Before training the models, the data was normalized by using sklearn's Standard Scaler.

8.2 Data set source analysis using LSH

8.2.1 Similarity of content across political spectrum. What this ultimately shows is that there are a lot of common terminology between the sites. 1 word ngrams basically show the presence of individual words between sites, and at the high mismatch rates found in N value of 1 shows that both the left and right leaning partisan discourses in

hyper partisan american-centric media share a lot of common terminology. As per our findings with N value of 2, two word phrases are also somewhat, although to an exponentially smaller degree, common occurrence among sites across polarities (24% mismatch across 364 match occurrences). When we get to 3 word phrases (N = 3), the results tighten up considerably, both in match occurrences and accuracy. 3 word phrases represent mostly complete ideas as opposed to terminology instances, so mismatch occurrences in this category would naturally indicate significant similarity metrics, and thus a causation for further investigation. MC values of 500 resulted in 11% mismatch and MC values of 1000 resulted in 4% mismatch. This latter result was seen as significant enough for further manual investigation.

From this starting perch, some curious results were gleaned. Less significant was that common mainstream sites like MSNBC was labeled as left leaning, and still had a match in the aforementioned key category with traditionally more right centric sites such as Newsmax. It is of interest too that mainstream sites such as MSNBC was labeled as left at all, especially by a fellow left identifying site such as BuzzFeed, which did the labeling. This seems to indicate that mainstream 'left' sites have a lot of content overlap with 'mainstream' right identifying sites, suggesting a more centrist, and therefore overlapping message structure across the american political center. More significant findings were found in the case of sites further down on the ends of the political spectrum, where sites like tomdispatch.com and activistpost.com were found to have sufficient similarity to be flagged with more extreme right leaning sites such as iotwreport.com The extreme left and extreme right therefore also seem to be sprouting similar messages.

8.2.2 Mislabeling prevalence. This necessitated the need for further investigation, so these sites were manually reviewed. One interesting case study that shall be discussed is of the site tomdispatch.com. In reviewing the texts of this site, it was found that, though it was a nominally left leaning site at initial glance, this site espouses a large degree of anti government rhetoric, something traditionally associated with right wing politics and 'small government talking points', in American political discourse. Democratic presidents were surprisingly criticized, although in view of not having done enough or even of doing the wrong actions altogether. This is in contrast to the other mismatched left-labeled site, the activistpost.com. This site sprouts decidedly anti-government rhetoric, espousing views that markedly seem anti-liberal in some senses. Its most vehement stance is in the necessity of freedom, and criticism of the stripping of personal liberties that are being undertaken, but by both parties that occupy the di-cracy of American political power. An interesting note is also the fact that it has several articles dedicated to attacking and deconstructing more mainstream but more clearly left-aligned news sites such as Vox and Vice News using language that more right-wing sites traditionally would. This site seems to defy conventional labels of left/right american politics, and begs the question of whether the manual labeling act was done correctly at all, or even it could have been for such a divisive outlier.

8.3 Choice of models

The base model for this project was a basic Artificial Neural Network (NN). As there is packages for adding layers and units for NN in Python this is easily implementable.

After finding the strong difference in phase on the reshaped datasets, as seen in 5 Pre-Processing, it was natural to try classification by Convolutional Neural Network(CNN).

Training a Support Vector Machine (SVM) for classification done using automatic gamma selection is simple to implement. As there is no tuning the method gave a accuracy of 0.7242 on the metadata.

In the text classification tasks, the words are considered as sequential data points. Thus, LSTM is efficient of processing these sequences. It would be clear that LSTM would perform better than other neural networks, as the model updates hidden layers for each timestamp and shares information of previous inputs to generate the next output of the sequence.

The last technique tried was AdaBoost. By performing AdaBoost it would be clear if the use of deeplearning would be necassery to perform the classification, or if a more energy efficient and faster shallow learning algorithm would be sufficient.

8.4 Tuning the models

The different models have different tuning parameter/hyperparameters. In this project two auto-tuners have been used, SigOpt and Keras Tuner. SigOpt was used for tuning AdaBoost, it was simple to implement and provided great graphical tools. It also provided a parameter importance score emphasizing the parameter with the most influence on the peroformance. This give valuable insights in what to prioritize when tuning the model. While doing hyperband optimization with Keras Tuner the min-max for number of units in each layer must be set. We did not find any improvement in performance by increasing maximum units from 100 to 520.

During manual architecture search for the NN and CNN we found that increasing the number of hidden layers would only make sense if the number of units in each layer was reduced at the same time. Often the results was comparable with an architecture without hidden layer, but increased number of units in the input layer. Working with the CNN on such small images (17x4, 32x16) made MaxPooling, ZeroPadding and increased number of convolution layers unnecessary. The only result was longer iteration time, without increased accuracy. We found recommendation of choosing size of input layers in the range between 1/2 - 2/3 of number of parameters. This did not increase performance for the NN compared to 30 units.

9 CONCLUSION

9.1 Veracity of BuzzFeed's labeling

It is seeming in conclusion that there is a great overlap in talking-points among similar albeit opposing strata of the political spectrum, and seeming in addition that some sites defy conventional points in the spectrum altogether. It may also be surmised that there are potentially some mislabeled sites, due to the great overlap in content data-points of significance with sites of opposing labels, which is of significance when taking into account our pre-existing dataset. These all add categories of dimensions in understanding the quality of the dataset BuzzFeed has produced. In conclusion however, it

can be surmised that a low enough level of mislabeling or labeling-challenged websites are present for it to be considered insignificant on the broad scale, but significant for maximum optimization levels (mislabeling percentages in low single digits with string stricter LSH parameters).

9.2 Pre-Processing

The integrity of the dataset is compromised by the apparently un-automated process of metadata collection. The large number of NaN values, especially in the fields: 'about', 'facebook_id' and 'website'. These were critical in cross referencing across other key datasets, rendering a large number of sites unusable; for example, almost 35% loss factor in Facebook statuses to Website political alignment cross referencing. The dataset is therefore less stringent than nominally apparent. The organization is also sub-optimal due to redundant multiplicities of data being present across several datasets. The sizes of the primary datasets are also lacking, and would provide greater usefulness in larger sizes.

9.3 ML Methods

One key takeaway from the implementation of multiple ML methods was the importance of Data Quality. Data quantity in critical instances did not return the ROI that was initially expected. A sample of 10,000 random points versus one of the full 4 million points on similar neural network architecture returned just 3% increase in performance. One thing to note however, on the smaller sample data set, the difference between the test and training sets were more pronounces, wherease in the larger (4 million) main data set, the discrepeancies were less noticeable due to a larger density of data allowing for less fluctuations and stability issues.

Since the task in question is primarily focused on hyperpartisan labeling, there are certain categories of data that falls under the potential moniker of bad data. Data near the centre with such generic text that almost defy labeling often existed, which were so nebulous they could easily fall into either categorization. On the other ends of the spectrum, there were often data that was so extreme, as to contain facets that could also be seen as belonging more appropriately to the other side. This was a particularly frequent case seen on the left end of the spectrum, as was found during the LSH investigation of data set veracity, where extremely hyperpartisan left sites would in fact end up attacking other left-leaning sites in many a manner right- leaning sites did. This causes degradation in the data set purity, which in turns reflects in its accuracy metrics.

It is therefore more pertinent to ensure data quality in the pre-processing stage, by filtering out indeterminate data types and piping in relevant data for maximum operational efficiency. Parameter optimization is also a key facet, as correctly optimized parameters are often more responsible for correct prediction capacity as opposed to more data under less ideal parameters (as seen in the LSH bucketing tasks). This is because ideally optimized parameters form the data for analysis in the best possible formats, and this task section also occupies a large amount of effort in formulating the ideal model.

In essence therefore, is key takeaways to optimize parameters correctly and pipe in better quality data, which also offers better energy use reductions. It is still of note that most of the different models had similar accuracy performances when working on the same cleaned dataset, which indicated that any of these models provides equal utility and usability and could be used interchangeably for this purpose.

10 FURTHER WORK

For future expansion on this work, an advantageous area of improvement would be to have a bigger metadata set to work on. However, due to the manual nature of the labeling effort, this might be unfeasible with the resources the source project started off with (research department of an entertainment site). This does however provide grounding for valid, more resource intensive research, and could be expanded upon to produce larger data sets for analysis.

The research could be expanded to automate the task of this aforementioned larger data set, and provide basis for automated labeling over much larger data sets found on sources like the Common Crawl, where petabytes of web data are stored over monthly crawls. In doing this, another key area of further work could be to make a more multidimensional polarity scale, which would take into account hard-to-label sites that fall outside or intransigently within the traditional left/right scale. This would re-purpose 'bad data' under different labels for other uses are insight. One way this automated task could be done could be through parallel processing on data intensive designed systems like Hadoop.

As valid for most machine learning project, the group was limited to minimal available compute power. The implications was restrictions on how big data sets could be run in some classifiers, both due to time constrains and insufficient memory. In a optimal circumstances, with unlimited compute, this project should be performed with one big data set that go thru the different pre-processing steps and classification algorithms. The data set should preferably contain both site metadata and status messages. This would make it possible to create a table of true performance comparison between the different approaches.

REFERENCES

James Chen. 2018. Neural Network. https://www.investopedia.com/terms/n/neuralnetwork.asp Last accessed 12 May 2022.

Definition.org. 2022. Definition of hyperpartisan. https://definition.org/define/hyperpartisan/ Last accessed 12 May 2022.

Akash Desarda. 2019. Understanding AdaBoost. https://towardsdatascience.com/understanding-adaboost-2f94f22d5bfe Last accessed 5 May 2022.

Faktisk.no. 2022. Fact checking site faktisk.no. https://www.faktisk.no/ Last accessed 12 May 2022.

Rohith Gandhi. 2018. Support Vector Machine — Introduction to Machine Learning Algorithms. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 Last accessed 13 May 2022.

Shikhar Gupta. 2018. Locality Sensitive Hashing. https://towardsdatascience.com/understanding-locality-sensitive-hashing-49f6d1f6134 Last accessed 12 May 2022.

Scikit Learn. 2022. sklearn.ensemble.AdaBoostClassifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html Last accessed 5 May 2022.

Long Ma and Yanqing Zhang. 2015. Using Word2Vec to process big text data. (Oct 2015). https://www.researchgate.net/publication/291153115_Using_Word2Vec_to_process_big_text_data

Marwa Naili, Anja Habacha Chaibi, and Henda Hajjami Ben Ghezala. 2017. Comparative study of word embedding methods in topic segmentation. *Procedia Computer Science* 112 (Sep 2017), 340–349. https://www.sciencedirect.com/science/article/pii/S1877050917313480

Michael Phi. 2018. Illustrated Guide to LSTM's and GRU's: A step by step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-

by-step-explanation-44e9eb85bf21 Last accessed 13 May 2022.

Sumit Saha. 2018. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 Last accessed 13 May 2022.

Anshul Saini. 2021. AdaBoost Algorithm – A Complete Guide for Beginners. https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/ Last accessed 13 May 2022.

Greg Corrado Jeffrey Dean Tomas Mikolov, Kai Chen. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of Workshop at ICLR* 2013 (Jan. 2013). https://www.researchgate.net/publication/234131319_Efficient_Estimation_of_Word_Representations_in_Vector_Space