Adversarial Normalization Method for Fake News Detection

PABLO KVITCA, Khoury College @ Northeastern University, USA

The popularization of social media and the democratization of the ability to cheaply broadcast information, deceptive content, including campaigns for intentional misinformation and disinformation, have become problematic. To keep the web a safe place, we require a scalable method to limit the spread of malicious content, and "fake news" in particular. The natural rapid change and rapid emergence of new topics around social media and news articles causes models to quickly become obsolete. This paper explores a Fake News Detection using a method base on BERT and Adversarial Neural Networks for normalization against learning features to specific news topics/events. We compare this to a non-adversarial model and analyze performance results and bias on key topics.

CCS Concepts: • **Computing methodologies** → *Neural networks.*

Additional Key Words and Phrases: fake news detection, natural language processing, neural networks, adversarial

1 INTRODUCTION

The democratization of the power to share and broadcast information through the Web has given masses of people the ability to share their stories and opinions. This has caused several issues around the reliability of information (and sources) being spread. There is an enormous interest in the ability to control public opinion, which has lead to problematic pandemic amounts deceptive and misleading content [13].

The problem is often characterized as "ever-growing" [9] plague [1]. Research on the topic has become more popular as the effects of misinformation became evident repeatedly during the USA's 2016 presidential election [1], and around other events throughout Europe, Latin America, Asia, and most of the world [2].

Currently, there are various approaches to improving the usability of Fake News Detection models, based on Natural Language Processing or Social Networks and using Content Metadata or Social Metadata [1], [8], [15]. These aim to use incorporate features and learn parameters that will help detect a news article as fake. One of the main challenges on this task is the rapid emergence of new topics and classes that are significantly different from previous content. This causes Machine Learning models to quickly become obsolete, which would require expensive manual labelling of new data and re-training of the models. In addition, the process of manually labelling articles and content as "fake" (or possibly more granular labels [17]) is time consuming. It seems improbable

Author's address: Pablo Kvitca, kvitca.p@northeastern.edu, Khoury College @ Northeastern University, Boston, USA.

the annotators would be able to keep up with the creation of new data. There are at least two ways to deal with this: First, we could use only external factor to the news content as features (including: historical reliability of the author, metrics on reactions and comments, social network graphs, etc). Second, we could tune an NLP model to ignore features of the groups of current data, aiming to, ideally, give importance to features that more directly relate to the deceptive nature of the content.

This project, inspired by the work on [18] which presents EANN (Event Adversarial Neural Networks for Fake News Detection), explore the second option: train an NLP model normalizing against the ability to detect specific known subgroups in the data. The method we used is based on the one proposed by [18], however there are some key differences with our model. First, EANN uses short text data (from Weibo and Twitter), in Chinese language. We chose to instead try to apply the method to longer text samples, with content from English news sources. Additionally, the data used for EANN has been previously labelled for "events", which are used as the adversarial feature on the model. The availability of datasets labelled both for "fake news" (or reliability) and events is limited, and we decided to use keywords of each news article, which we consider more accessible.

The report first defines the general problem of fake news and the scope of this project. We then go on to describe the data used and the necessary pre-processing. Following, we describe the architecture of the the Adversarial Neural Network model, with the opposed (adversarial) training objectives), and the methods used to incorporate BERT for token classification. We present the results on the testing data, and due to the sentive topic of Fake News Detection, go on to analyze and compare the error and performance of the model that could show bias for or against specific topics. We then discuss the results and future potential steps for the project.

1.1 Accompanying Jupyter Notebooks

Please note the three accompanying Python Jupyter Notebooks that show the data cleaning and pre-processing (*00.preprocessing-cleaning.ipynb* and *01.pre-make-full.ipynb*), and the model implementation, training & result analysis (*02.final-adv-fann.ipynb*), respectively.

1.2 Defining Fake News

While "fake news" have existed for centuries (at least since the invention of the printing press, as explained in [13]), it the World Wide Web has enable massive systematic misinformation campaigns, leveraging the inter-connectivity of social media, "echo chambers" (or "bubbles"), "viral" content [13].

Most research agrees on a basic definition for "fake news": **content that aims to deceive using false claims to steer opinion towards a useful point of view**. However, we must not forget that "fake news" is highly complex topic and touches on the philosophical concept of “truth“. This project explores the technical implementation and potential performance gains from using an adversarial network for normalization. It does not go into the complex and nuanced tangents of: which content should be considered "fake" or "real" or whether the author’s intentions are to be taken into account. The project is based on a pre-labelled dataset and assumes the dataset’s labelling to be reliable and accurate. The goal is to explore the adversarial topic detection approach, which could potentially be applied to different datasets.

The power to skew public opinion can gave devastating effects, causing real damage at a global scale. It could lead to actors maliciously influencing global and national politics, trade markets, and causing violence. The stories and news content can "go viral" fast and be shared countless times within seconds. This all demostrates a rising need for scalable models that can detect and classify fake news in a timely fashion.

2 MODELS & EXPERIMENTAL SETUP

The aim of this project is to explore whether applying an "adversarial normalization" technique to a Neural Neural network can improve its performance on unseen cases and reduce the number of mis-classified documents (false negatives, false positives), both overall and on sub-groups that could be considered sensitive to bias. In this section, we first describe the used dataset and explain the applied pre-processing; we then continue to show the details of the Adversarial Network, its architecture, and implementation. Finally, we also describe the experimental setup used for comparing the models.

2.1 Data and Pre-processing

The data used by this model requires two sets of labels for each document. First, a label for the "reliability" of the document is needed. This could be a binary label (`real` vs `fake`) or multi-class (for example, the popular LIAR dataset [17] uses the labels: `true`, `mostly-true`, `half-true`, `barely-true`, `false`, and `pants-fire`). Second, we require a label that describes groups in the dataset that can be used to normalize against. On the EANN model, [18], this second label represents the `events` the content is discussion. In our dataset we use a computed `keyword` label extracted from the data, as explained in the **Dataset: FakeNewsCorpus** section below.

The model used in the project is based solely on text features (mainly the `title` and `content` of a news article), as opposed to the EANN model which used multi-modal features from image and text combined (see [18]. We decided to focus on the text and Natural Language Processing features in order to: simplify the model, reduce data requirements, and reduce training/evaluation time. However, the main goal of this approach is to use an adversarial model to improve performance and re-usability, and reduce bias. The each of the components (the base processing and the two classification heads, for fake news and adversarial features, respectively) could be replace with different architectures. This would allow the model to: incorporate different types of features (such as images or social metadata); use different adversarial classification classes; and/or use different architectures for Natural Language Processing.

2.1.1 Dataset: FakeNewsCorpus.

The dataset we selected for the project is `FakeNewsCorpus` [14]. It is freely available (under the Apache License 2.0), and contains data scrapped from OpenSources [10] and The New York Times [16]. Each document has one of 10 `type` labels: `fake`, `satire`, `bias`, `conspiracy`, `junksci`, `hate`, `clickbait`, `unreliable`, `political`, `reliable`; however, we merge these labels into binary `real`/`fake` classes for simplicity, as explained below. The complete dataset has around 9 million sample, however some of these where either wrongly formatted (or non-specified format), had where duplicated, or had key features missing. After cleaning unusable data, only around 1 million remained usable.

The dataset has various features for each document, including the: source `domain` and `url`, timestamps (`scraped_at`, `updated_at`,`inserted_at`), the document's `title` and `content`. As well as the content `type`, a list of `authors`, a list of `keywords`, and a list of `tags`.

2.1.2 Pre-Processing.

The initial pre-processing of the data involved loading and analyzing the available data. We cleaned and parsed each of the `keywords`, `tags`, and `authors` fields, for each of which we counted the number of documents they appeared in and created a sorted order for the 100 most common (respectively). Later, we turned each of the most common labels into binary columns, each document ended up with 300 new columns, which had a hot 1 value if that document original had the corresponding `keyword`, `tag`, or `author`.

Since we require the keywords label for the adversarial classification, we generated another column, called `keyword`, representing the document's single keyword that is part of top 25 most common keywords. Example: if the most common keywords are "climate change", "politics", and "business", then a document with only the keyword "politics" will select that as its main keyword; a document with "climate change" and "business" as its keywords will select "climate change" for its main keyword.

This process allows us to have a single keyword class per document, which we can use for adversarial training. We then dropped all documents that had none of the top 25 most common keywords. This was done to reduce the total number of classes possible for the adversarial training.

Note that the Adversarial Classification head (see **Models Setup & Training**) could be modified to support multiple labels per document. We decided to keep the model to single labels with just a few classes to reduce the complexity and training time for the model. An exploration of using multi-label classification here would be worthwhile.

To further simplify the model, we also decided to merge the 10-fold `type` labels into binary classes. This increases the balance between each class and reduces the amount of encoding needed. We chose to arbitrarily consider the types `conspiracy`, `fake`, `unreliable`, `rumor`, `clickbait`, `junksci`, `satire`, and `hate` as the `FAKE` class, and the `unknown` and `reliable` types as the `REAL` class. There are, of course, different choices that could be made here. Again, note that the Fake News Classification head (see **Models Setup & Training**) could be modified to use all 10 classes.

The last step in the pre-procesing involved cleaning and joining each line in the `content` of the documents, changing the new-line-character for ¨
nänd joining it to the document `title`, separated by an "endtitle" token. This is done to keep a single text input for the Neural Network. However, the base networks steps could be modified to take separated inputs. Lastly, we measure the total tokens for each of the resulting documents, looking for a suitable `max token length` to use on our model (we chose truncate or pad all documents to 512 tokens).

The resulting cleaned, trimmed, and pre-processed dataset had just around 40,000 samples. We had originally looked for large datasets but we are using a small section of the data for training to speed up the training process. The distribution of classes in the resulting data was around 60% for fake samples and 40% for real samples, as shown below on Figure 1.

We created two versions of the dataset, one with the resulting proportions between classes(as seen on Figure 1), and one artificially balanced have exactly 50% of each class. After some testing, both versions performed similarly and the balanced version was discarded to reduce the number of models needed to train.

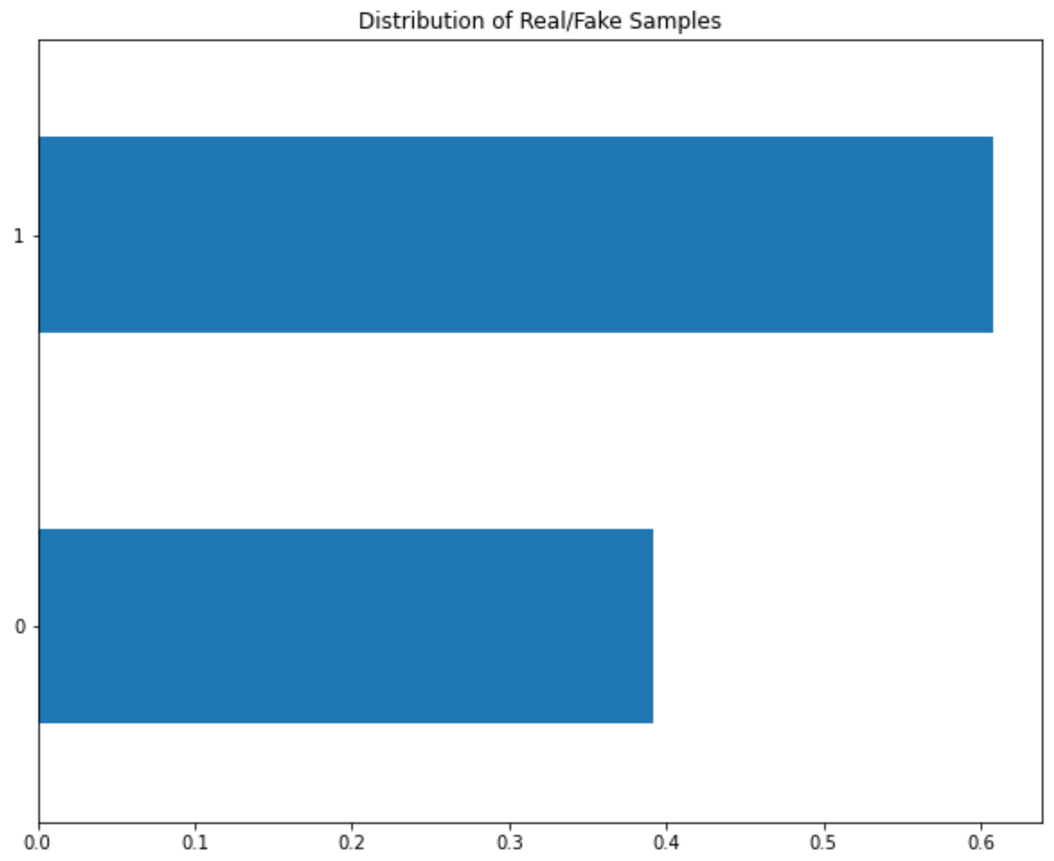

Fig. 1. Resulting Class Distribution on Fake/Real Classes.

2.2 Adversarial Neural Network for Normalization

The most common application for Adversarial Networks is automatic generation tasks (see [6] and [19]), since they allow for a much faster turn-around during training (compared to manual checking of the outputs generated). These architectures use two Neural Nets with opposing (adversarial) goals as a type of min-max game [18]. The setup on this project, however, aims to improve the performance for unseen samples when new topics (or types/events) are expected to show up on the data, this is seen as a type of "normalization" against the classification target of the opposing model.

As proposed on [18], the Event Adversarial Neural Network (EANN) uses the intuition that if the adversarial classifier can determine which event was on the input for the model based on the output for the last layer, then the model is learning event-specific features. Similarly, our model tries to determine the *topic keyword* of the original document form the last layer output. We setup an "adversarial game" between 2 classification targets, referred to as "heads": first a **Fake News Classification** head, and second an **Adversarial Classification** head.

2.2.1 *Adversarial Classes.*

As explained earlier, the model requires 2 labels for each document: first, the main target (in this case the `real/fake` labels); and second, the adversarial target. Since we lacked the "event" labels (used in EANN), we decided to use the main `keyword` classes as our adversarial target. This was convenient from the data on the selected dataset, but we could have similarly chosen the main `tag`. Other datasets may require extra annotation or scrapping to acquire the second label, but some sources have easily accessible labels. For example, Tweets maybe be annotated with their hashtags or associated "trending" topics.

2.3 Models Setup & Training

We used the library `PyTorch` [11] to implement our model with HuggingFace's `transformers` [5] library for pre-trained BERT (an updated `PyTorch` compatible model based on the original *BERT pre-training of Deep Bidirectional Transformers* [4]).

We created a custom `Module` that allows us to easily turn on/off the adversarial section of the overall network. Later, we use this to create 2 variations of the trained models. The custom module (named `FANN`) initializes and pre-loads the data for producing the BERT Embeddings, and initializes the layers for: the *base processing*, the *Fake News Classification* head, and the *Adversarial Classification* head (if enabled).

On all models, the input is first processed by BERT's Embedding layers. The custom module allows for choosing between using BERT's pre-pooled classification token embedding, (see **BERT Embeddings** section) or using embeddings for the whole input. Using all the embeddings proved too slow for training due to the large dimensions needed (`embedding size` * `input tokens`). For our testing, we used only the pooled version. The output is then passed to a `Linear` layer with a `LeakyReLU` activation function [12] (selected with trial-and-error).

After the *base processing* stage, the layer output is used by the classification heads. The *Fake News Classification* head applies another `Linear` layer with a `Softmax` activation for the final classification. If enabled, the *Adversarial Classification* head simultaneously takes the base out into a separate `Linear` also using a `Softmax` activation (this times with the adversarial classes as outputs). During training, the adversarial head also applies an "adversarial negative function", this serves the function of opposing (taking the inverse) the accumulated loss to use as a normalization for the adversarial head. Figure 2, below, shows a graphical representation of the network architecture.

2.3.1 *BERT Embeddings.*

We chose to use the base BERT word embeddings model to numerically encode the text on our data. The combined `title` and `content` of the documents is fed to the BERT layer. In short, this layer encodes the specific task context "bidirectionally" which eventually results in an output layer, while fine-tuning the all the parameters of the pretrained Transformer for the current context [20] ([3]).

The first token for all outputs of the embeddings layer is a special *classification token*) (`<CLS>`), that represent pooled embeddings of all other tokens. On our tested models, we use this as input the the next hidden layer.

2.3.2 *Fake News & Adversarial Classification Heads.*

Each classification head has a simple short architecture, though the *Adversarial Classification* can be disabled. Both take the same output of the *base processing* into their own `Linear` layers with

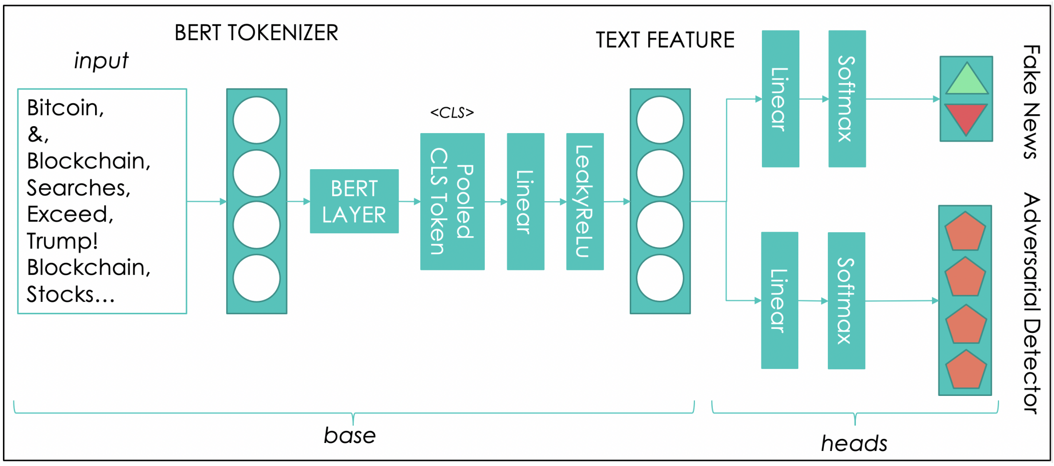

Fig. 2. Adversarial Fake News Detection Neural Network model Architecture Diagram

a `Softmax` activation. Here the *Fake News Classification* head uses the binary classes as outputs, while the *Adversarial Classification* head uses all of the "adversarial" classes.

During the training backward pass, we take the opposite of the loss accumulated from the adversarial layer to shift the network parameters against the adversarial targets, (but still towards the main targets).

2.4 Experiments & Metrics

To compare the overall resulting performance between using an adversarial model or a "simple" model, we setup 2 separate models for our testing, one with an enabled adversarial head and one disabled. All other hyper-parameters and functions (learning rate, batch size, hidden layer sizes, pre-trained BERT version, and input token lengths; and optimizer and loss function) are the same on both models.

We split the data available into 3 sections, a *training set* with 12,000 samples, a *validation set* with $\tilde{3}$,500 samples, and a *test set* with $\tilde{3}$,900 samples. We then trained both model using the *training set*, and *validation set* for periodical evaluation, for 10 epochs. Leaving the *test set* untouched.

After training, we evaluated each model on all samples (keeping the *training*, *validation*, and *test* sets separate). We computed various metrics scores for each set on each model, showing **precision**, **recall**, **accuracy**, and **f1** scores, as well as the **confusion matrix** and the **ROC Area under Curve**.

For analyzing and comparing the potential bias of the models, and the effect of the adversarial approach, we separated samples associated to some of the most common keywords and some of the most common tags. These where arbitrarily picked from the 25 top keywords and 25 top tags, respectively. Then, for each picked keyword/tag, we split the *test set* data into a group **without** the relevant sample and a group of **only** those samples. Finally, we computed **precision**, **recall**, **accuracy**, and **f1** scores, and the **ROC Area under Curve** value for the full *test set*, and each group. We specifically use the **ROC Area under Curve** [7] value as it relates the sensitivity (True

Positive Rate) and FPR (False Positive Rate, in turn relating to the specificity, True Negative Rate). We use this value to guage the improve on mis-classification for the selected keyword/tag (based on false positives and false negatives).

Some of these results are shown and cited in the **Results** section, but please referred to the Jupyter Notebook for further details and plots.

2.4.1 Selected Hyper-Parameter Values & Functions.

The selected hyper-parameter values and functions were chosen arbitrarily and through trial-and-error, eventually settling on a set that resulted on reasonable performance on the *training set* and final loss during training, with the trade-off of computation time needed to complete each training epoch. The resulting hyper-parameters were: `Input tokens`: 512, `pre-trained BERT model`: 'bert-base-cased', `Used pooled BERT output`: true, `Learning Rate`: 1e-06, `# Epochs`: 10, `Batch Size`: 8. In addition, both models also used an *Adam* `optimizer` and a simple *CrossEntropyLoss* loss function as the `criterion`.

3 RESULTS

The resulting scores on for each model on the *training*, *validation*, and *test* sets show overall improved performance by using the adversarial approach. Similarly, most of the selected keywords/tags checked for false positives/false negatives seem to have improved error rates on the adversarial model. We focus on the `accuracy` and `f1` scores for comparing overall performance, and on the `ROC AuC` value for analyzing the error for bias for/against the selected keywords/tags.

3.1 Overall Performance

We analyze the `accuracy` and `f1` score improvements on the data between the *simple* model and the *adversarial* model. The adversarial model seems to have a noticeable effect on the performance. The *adversarial* model performs slightly better: showing an improvement of $\tilde{2}$% for both seen and unseen samples, versus the *simple* model. Figure 3, below, shows two graphs: we display `accuracy` (left) and `f1` scores (right). For each graph, we show the scores for each data set (*training*, *validation*, and *test*). The *adversarial* model is shown in `purple` and the *simple* model in `orange`. We consider these results to show the model is starting to learn features independent to the adversarial targets, but it could be improved with more data and epochs, and hyper-parameter tuning.

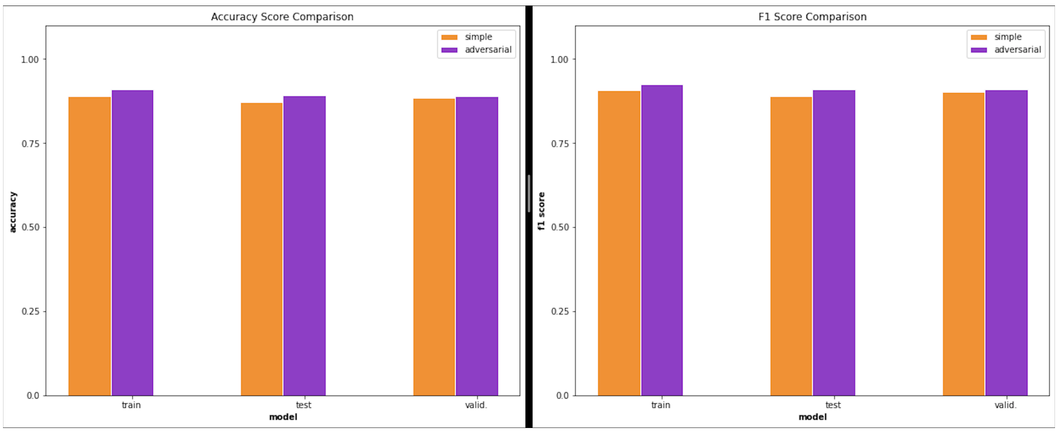

Fig. 3. Overall Accuracy & F1 Scores Comparison between Simple and Adversarial models

3.2 Exploring Bias on Topics (Keywords/Tags)

All computer-based decision tasks and Machine Learning implementations that could be potentially applied to the real world have various ethical and societal implications that need to considered. Applications of Neural Networks, in particular, must deal with ensuring fairness for protected groups (and other factors) and possible explainability. The task of Fake News Detection is furthermore concerned with the need to protect critical human rights (free speech) and prevent censorship and bias.

We focus the bias exploration mainly on the error rates for false positives and false negatives, through the `ROC AuC` value, because we believe it captures the two most relevant factors for bias. Both mis-classification types can have huge repercussions: If a system like this is deployed and users grow accustomed to trusting its results, there is an increased danger from false labeling. A fake news story perceived (and marked) as real can cause more damage than leaving it unlabelled. Conversely, a label as a fake story could have the same effect and even hurt the credibility of entities mentioned, authors, publishers.

Here, to compare the classification between subgroups we score on the data from the *test set* in 3 ways: all the data, all data except the selected subgroup, and only the data of the selected subgroup.

3.2.1 Bias on Topic Keywords.

We compare the `ROC AuC` value for different keywords picked from the top 25 most common on the dataset. The first set compares 4 keywords on topics: *climate change*, *business*, *politics*, and *science*. In Figure 4, below, each graph shows the scores for selected subgroup *all data* (`orange`), *all except selected* (`purple`), and *only selected* (`green`) on each model, with the scores for the *simple* model on the left and the *adversarial* model on the right. See **Appendix A: Larger Plots** for prints of each graph individually. We observe a clear improvement on the error rates for 3 of the selected topics. However, the topic *science* shows a higher errors. We believe this might be due to the small number of cases for this topic (around 10 times less).

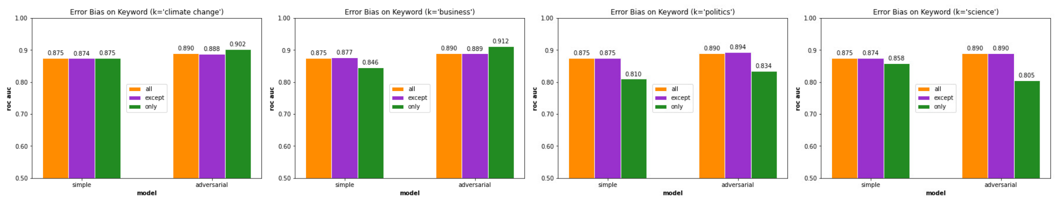

Fig. 4. ROC AuC Comparison for Error Rate Bias on Topics

3.2.2 Bias on Countries Tags.

We compare the `ROC AuC` value for 2 different tags picked from the top 25 most common on the dataset. These denote documents related to 2 countries: *israel* and *russia*. Note that the tags for a document could be multiple (rather than a single one for keywords), the same document could show up on both tags' subsets. Figure 5, below, shows scores the scores for *israel* on the left and *russia* on the right. See **Appendix A: Larger Plots** for prints of each graph individually. The adversarial model mangaes a huge improvement on performance for *israel*, $\tilde{2}8\%$. We consider this a promising result regarding the potential to improve the error rates for biased subclasses on dataset.

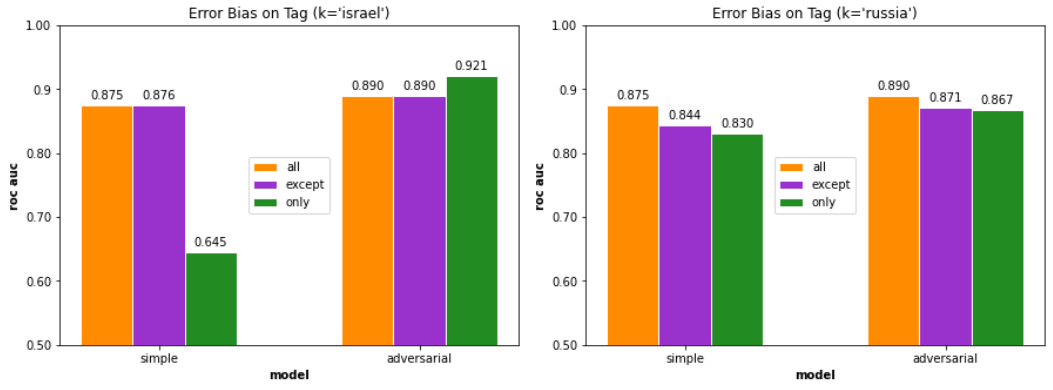

Fig. 5. ROC AuC Comparison for Error Rate Bias on Countries

3.2.3 *Bias on High-Profile Personalities.*

We also selected 2 keywords and 2 tags of High-Profile Personalities around news stories. Note that, as before, the keyword are a single one per document while the tags are not. This means documents with both keywords will only be counted for the most common keyword; while the tags' subsets will count all cases related to that tag. We selected the keywords for *barack obama* and *donald trump* and the tags for *donald trump* and *hillary clinton*. These news stories date from around the 2016 US Presidential Elections Campaign, and we believe can be used for interesting comparisons for bias between political parties. See **Appendix A: Larger Plots** for prints of each graph individually.

Figure 6, below, shows the graphs for each selected keyword and tag. In all 4 case, we observe an improvement on the error rates for samples related to those personalities. If we compare the to graphs for *donald trump* (middle two) we see there was a similar improvement for the classification, bringing it closer to the norm. The models, in general, seem likely to mis-classify content related to these high-profile individuals. This is expected since both real and fake news often mention politicians. It is important to note that in all cases, the *adversarial* model improves the error rates. We consider this to show a promising potential for the adversarial approach.

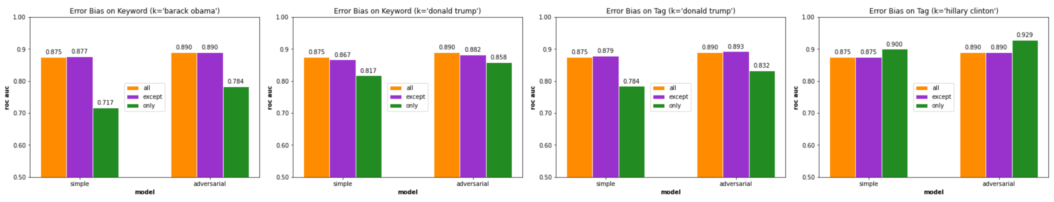

Fig. 6. ROC AuC Comparison for Error Rate Bias on High-Profile Personalities

3.2.4 *Ethical & Social Implications.*

Automated filtering of content by machines creates a very specific (and well-founded) fear. The right to freedom of speech has been essential for the development of current democratic governments and culture. In addition, those in control of a fake news detection system might unconsciously or consciously impart biases towards the filtering of news stories. The power of controlling, or at least influencing, this systems for own gain is dangerous, and worsened by the common and erroneous belief that machines only make objective decisions [2]. The analysis in this report only concerns itself with whether mis-classification error rates where improved by the adversarial model. In case of a system based on this model were to be considered for application on the real world, further research into it must be done, including more tests for classification bias, societal implications of the overall system, availability for new data, requirements for re-training the system, what actions should be taken on detected fake news, and many other factors.

Some of the factors for bias checking that should be considered could include preference towards or against: ideological movements, political parties, cultures, writing styles, protected groups, sources, languages, regions, organizations, public figures, individuals, publishers, and authors.

4 CONCLUSION & NEXT STEPS

In this report we introduced the task of Fake News Detection and motivation for its high importance. We presented the original EANN model [18] and described the differences to the model presented here. We observed overall improved performance on previously seen and unseen samples by using an *adversarial* model, to "encourage" the Neural Net to rely on more general and re-usable features.

We believe there is strong potential to applying an *adversarial* approach to the task of Fake News Detection. While the overall performance improvements were not significant, we think spending more time on fine-tuning hyper-parameters and allowing for longer training periods, combined with larger/better datasets, could show larger changes.

The exploration of potential bias for specific keywords/tags on the dataset seems to show equally promising results. In most of the cases, the *adversarial* model was able to improve its error rates. However, further analysis of this data on more keywords/tags is necessary to one approach is "less biased" than the other. We believe this method could be used to improve the error rates for known biases on the data.

As a possible next step for the project, we might consider changing BERT for state of the art word embeddings (like GPT-3) and fine tuning the hyper-parameters through testing, and also exploring different hidden layer architectures (like CNNs or RNNs). An interesting option could be to modify the adversarial detection head to use multi-label targets (removing the limitations of selecting the documents' most common keyword). Similarly, it could be possible to switch the fake news labels from binary to the original 10-fold types.

Regardless of next steps, the first step should be to search for alternative and larger datasets that could be used for the task (possibly scrapped Twitter data). The dataset currently used resulted in little usable data (though further work might be able to "recover" more usable samples).

Additionally, it would be interesting to train and measure the performance on the LIAR [17] benchmark dataset and to include a baseline text classification model (maybe based on Bag-of-Word with Naive Bayes) for comparison. Another potential step could be search for datasets on different languages (maybe Spanish), or multi-language datasets, to see if the improved performance is maintained on different document languages.

In conclusion, this method seems to be able provide good improvements to the classification of fake news; as it forces the Neural Net to rely on more general features; and exploring the potential next steps seems worthwhile.

5 ACKNOWLEDGEMENTS

We thank the support and help from Professor Byron Wallace and Northeastern University's Khoury College's Fall 2020 course on Practical Neural Networks (DS 4440); the book Dive Into Deep Learning [20]; and the developers and documentations for: NumPy, Pandas, Scikit-Learn, Matplotlib, PyTorch, HuggingFace's transformers, Seaborn, Python, Jupyter Notebooks, and Google Colab.

REFERENCES

[1] Gaurav Bhatt, Aman Sharma, Shivam Sharma, Ankush Nagpal, Balasubramanian Raman, and Ankush Mittal. 2017. On the Benefit of Combining Neural, Statistical and External Features for Fake News Identification. *CoRR* abs/1712.03935 (2017). arXiv:1712.03935 http://arxiv.org/abs/1712.03935

[2] Lia Bozarth and Ceren Budak. 2020. Toward a Better Performance Evaluation Framework for Fake News Classification. *Proceedings of the International AAAI Conference on Web and Social Media* 14, 1 (May 2020), 60–71. https://aaai.org/ojs/index.php/ICWSM/article/view/7279

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. [n.d.]. *Dive Into Deep Learning - 14.8.3. BERT: Combining the Best of Both Worlds.* https://d2l.ai/chapter_natural-language-processing-pretraining/bert.html#bert-combining-the-best-of-both-worlds

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[5] Hugging Face. [n.d.]. *Hugging Face's pretrained BERT Model.* https://huggingface.co/transformers/model_doc/bert.html#bertmodel

[6] Thai Le, Suhang Wang, and Dongwon Lee. 2020. MALCOM: Generating Malicious Comments to Attack Neural Fake News Detection Models. arXiv:2009.01048 [cs.CL]

[7] Scikit Learn. [n.d.]. *Scikit Learn User Guide - 3.3.2.14. Receiver operating characteristic (ROC).* https://scikit-learn.org/stable/modules/model_evaluation.html#roc-metrics

[8] Yi-Ju Lu and Cheng-Te Li. 2020. GCAN: Graph-aware Co-Attention Networks for Explainable Fake News Detection on Social Media. arXiv:2004.11648 [cs.CL]

[9] Kai Nakamura, Sharon Levy, and William Yang Wang. 2020. r/Fakeddit: A New Multimodal Benchmark Dataset for Fine-grained Fake News Detection. arXiv:1911.03854 [cs.CL]

[10] OpenSouresCo. [n.d.]. *OpenSources Co.* http://www.opensources.co/

[11] PyTorch. [n.d.]. *PyTorch Documentation.* https://pytorch.org/

[12] PyTorch. [n.d.]. *PyTorch Documentation on LeakyReLU.* https://pytorch.org/docs/master/generated/torch.nn.LeakyReLU.html#leakyrelu

[13] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake News Detection on Social Media: A Data Mining Perspective. *CoRR* abs/1708.01967 (2017). arXiv:1708.01967 http://arxiv.org/abs/1708.01967

[14] Maciej Szpakowski. 2020. Fake News Corpus. (Jan 2020).

[15] Eugenio Tacchini, Gabriele Ballarin, Marco L. Della Vedova, Stefano Moret, and Luca de Alfaro. 2017. Some Like it Hoax: Automated Fake News Detection in Social Networks. arXiv:1704.07506 [cs.LG]

[16] New York Times. [n.d.]. *New York Times.* https://www.nytimes.com/

[17] William Yang Wang. 2017. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers).* Association for Computational Linguistics, Vancouver, Canada, 422–426. https://doi.org/10.18653/v1/P17-2067

[18] Yaqing Wang, Fenglong Ma, Zhiwei Jin, Ye Yuan, Guangxu Xun, Kishlay Jha, Lu Su, and Jing Gao. 2018. EANN: Event Adversarial Neural Networks for Multi-Modal Fake News Detection. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining* (London, United Kingdom) *(KDD '18).* Association for Computing Machinery, New York, NY, USA, 849–857. https://doi.org/10.1145/3219819.3219903

[19] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending Against Neural Fake News. arXiv:1905.12616 [cs.CL]

[20] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. 2020. *Dive into Deep Learning.* https://d2l.ai.

A APPENDIX A: LARGER PLOTS

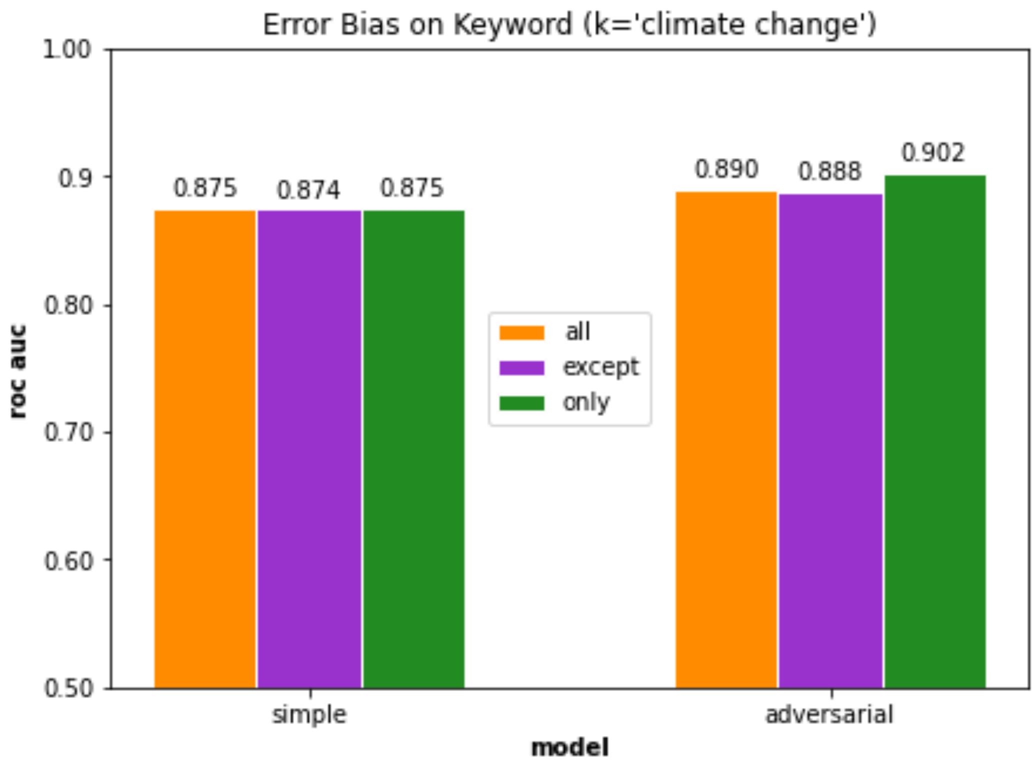

Fig. 7. ROC AuC Comparison for Error Rate Bias on Climate Change Topic

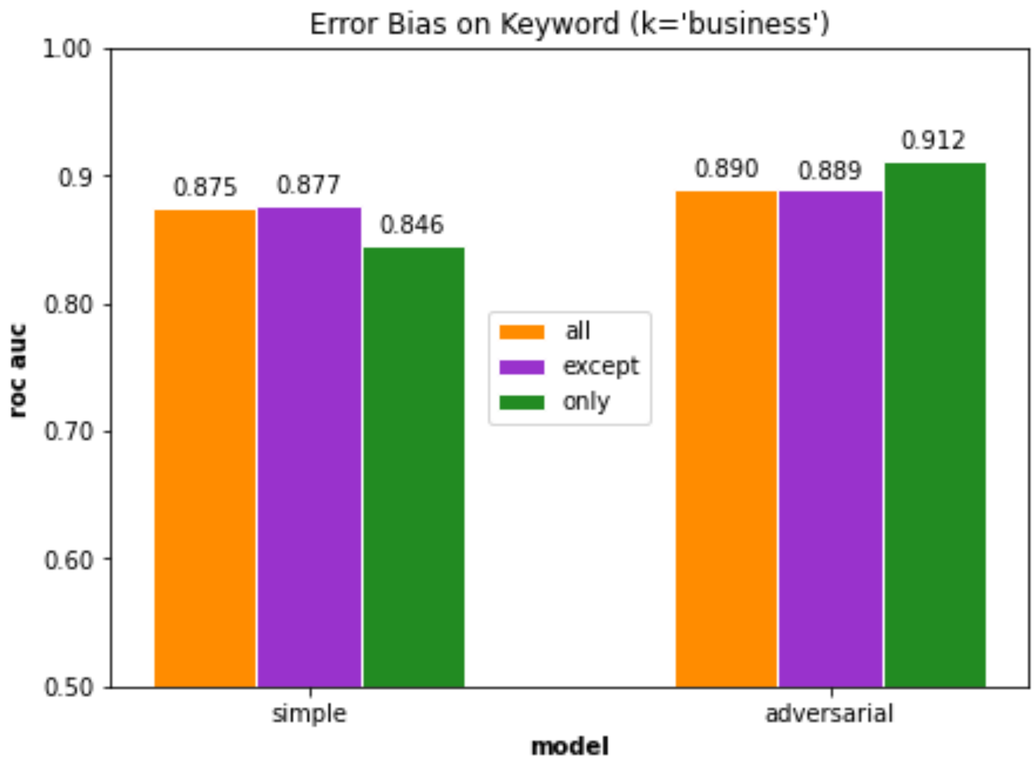

Fig. 8. ROC AuC Comparison for Error Rate Bias on Business Topic

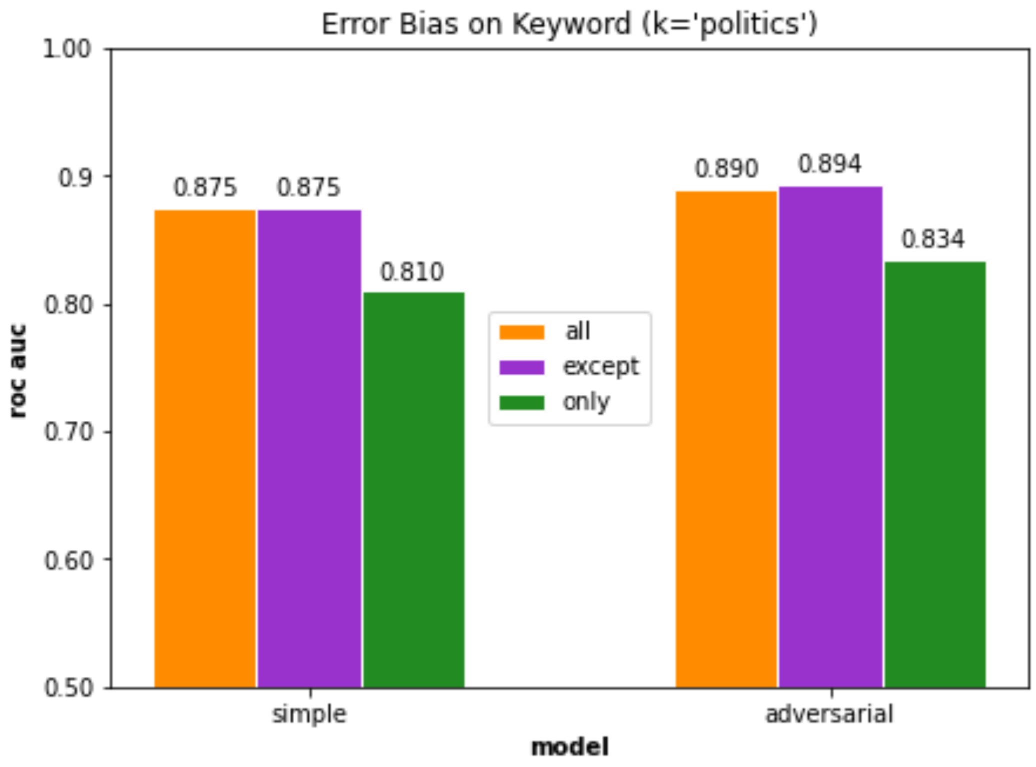

Fig. 9. ROC AuC Comparison for Error Rate Bias on Politics Topic

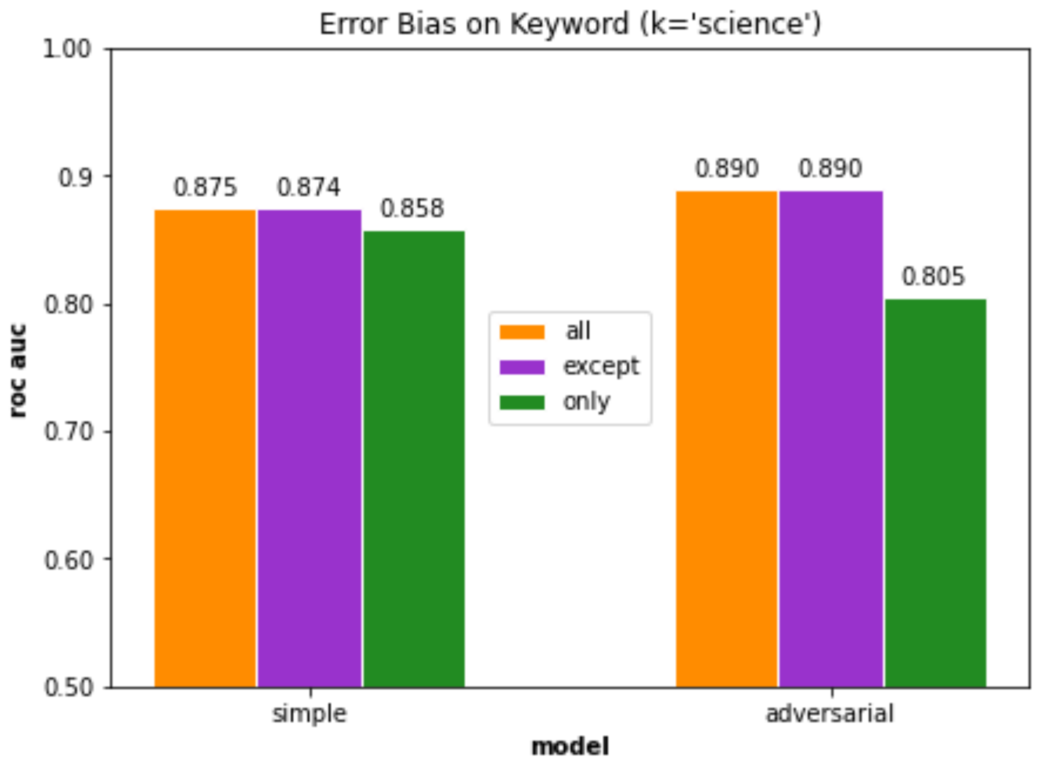

Fig. 10. ROC AuC Comparison for Error Rate Bias on Science Topic

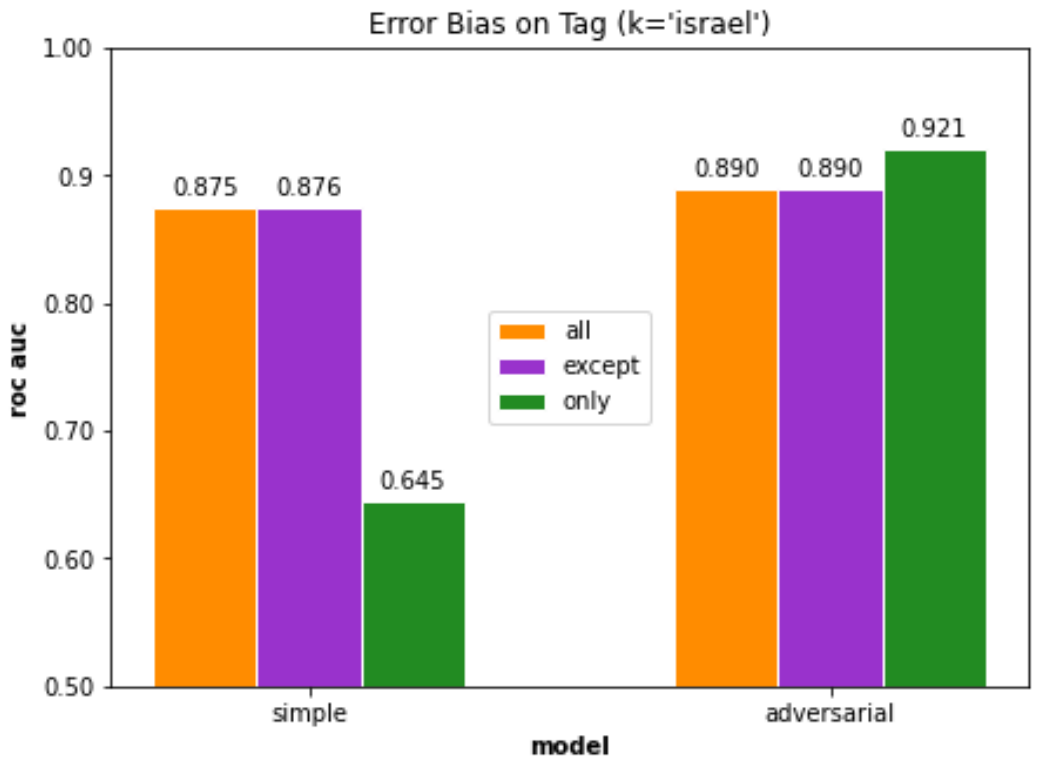

Fig. 11. ROC AuC Comparison for Error Rate Bias on Israel Country

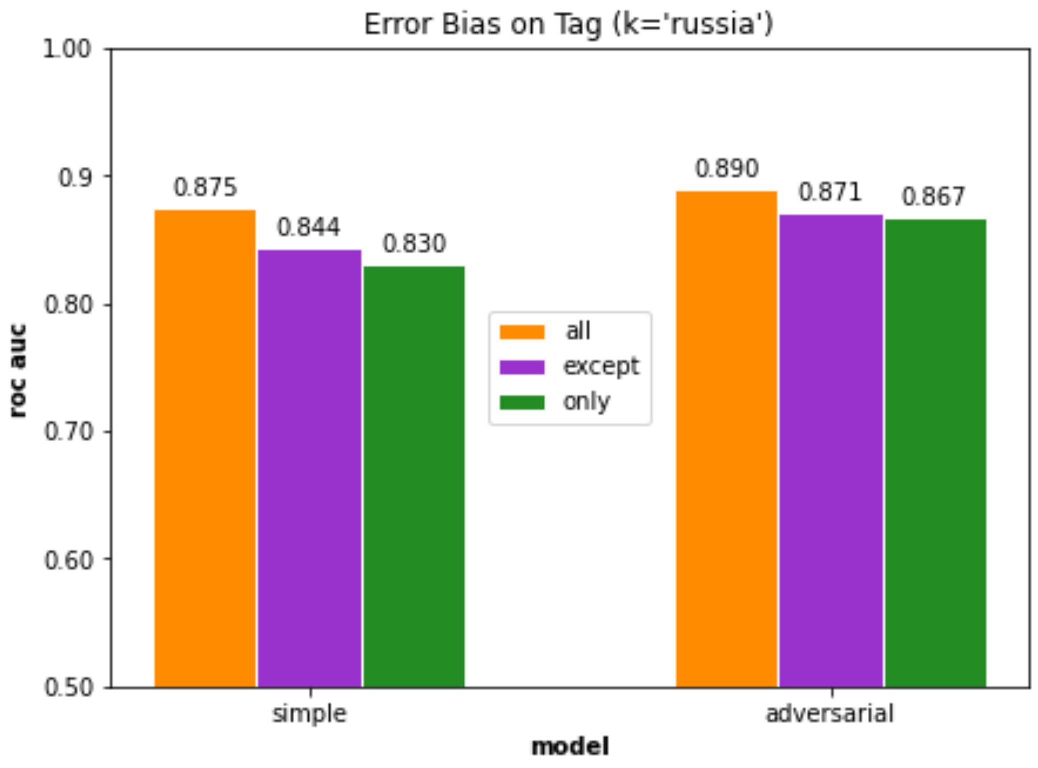

Fig. 12. ROC AuC Comparison for Error Rate Bias on Russia Country

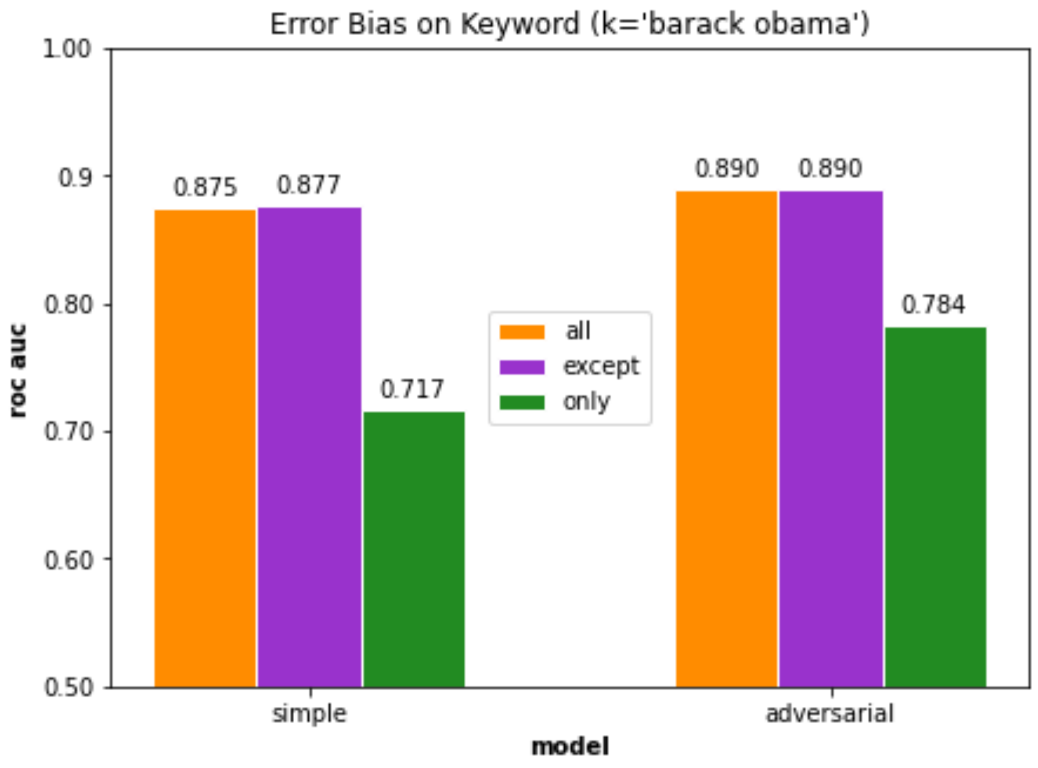

Fig. 13. ROC AuC Comparison for Error Rate Bias on barack obama

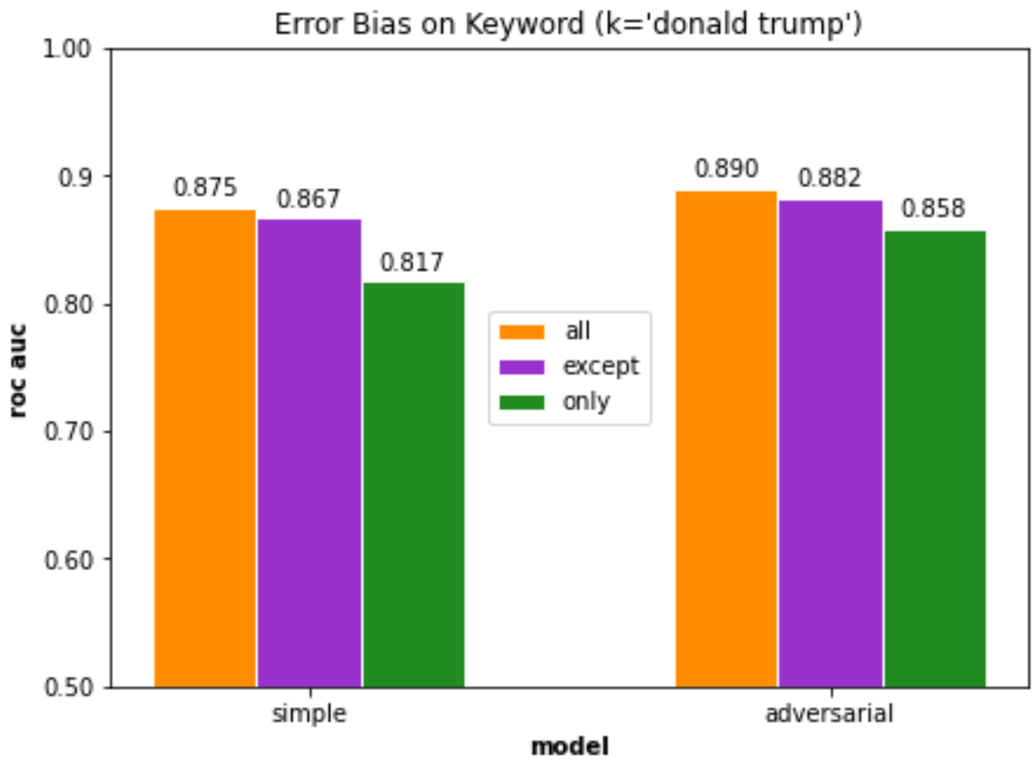

Fig. 14. ROC AuC Comparison for Error Rate Bias on donald trump

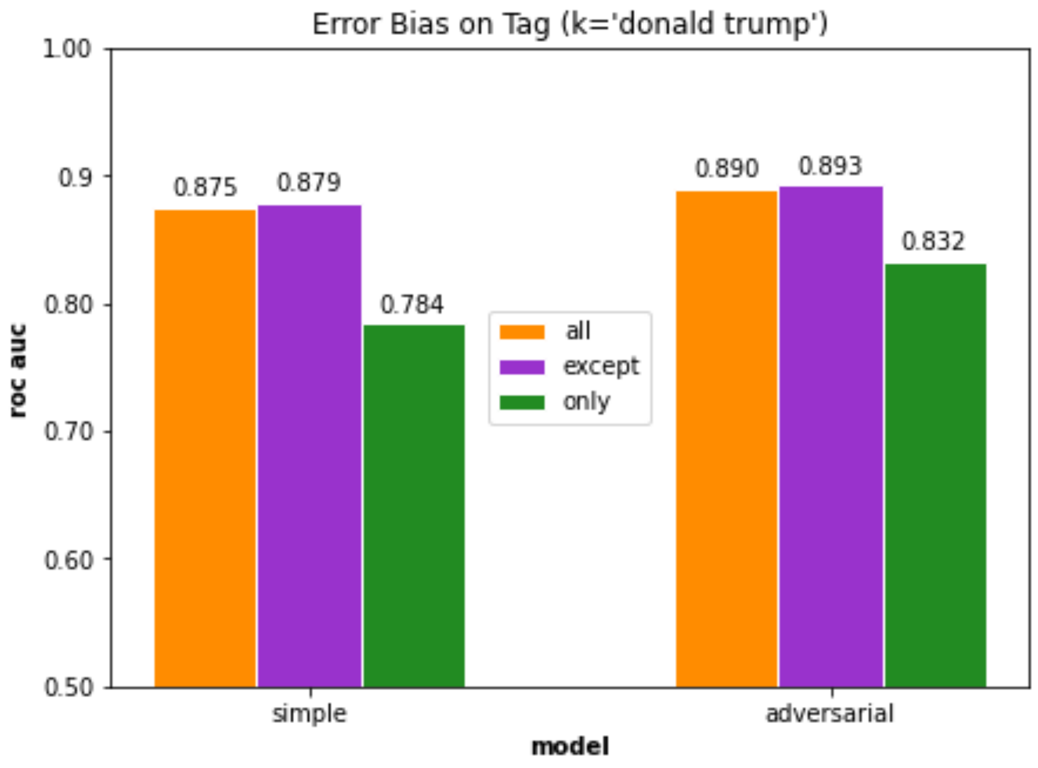

Fig. 15. ROC AuC Comparison for Error Rate Bias on donald trump (tag)

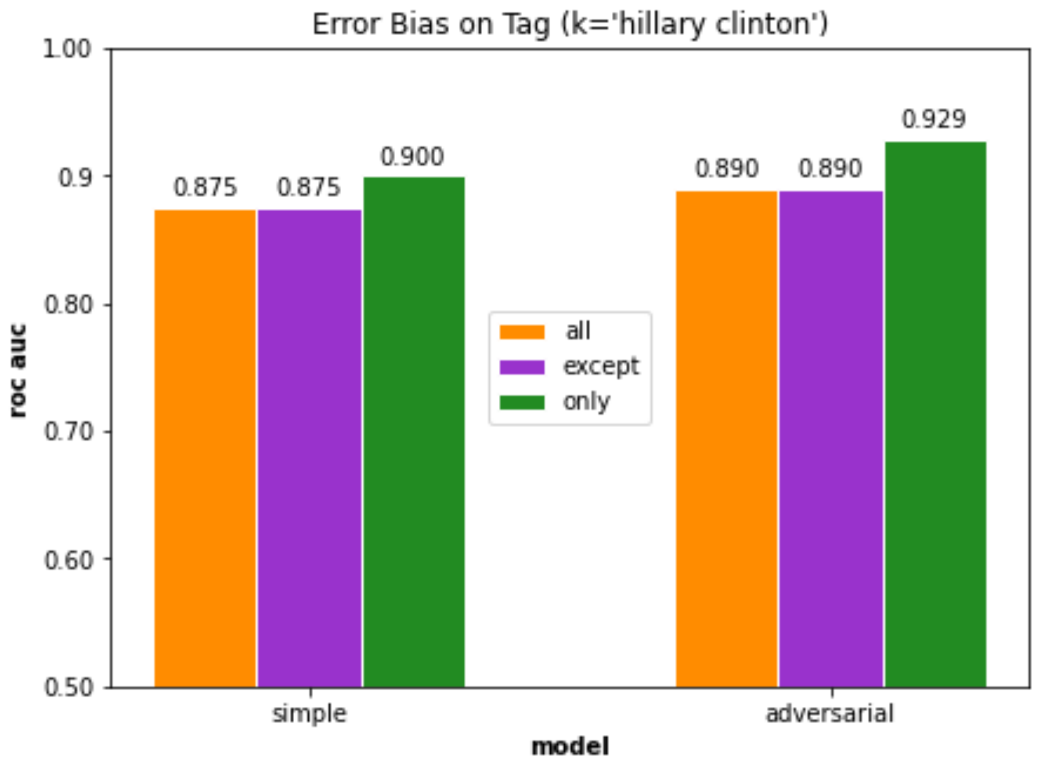

Fig. 16. ROC AuC Comparison for Error Rate Bias on hillary clinton (tag)