Group: Seize the Data

Large Scale Generation of Falsified Scientific Literature and Detection

Introduction

Creating falsified papers based on the original Bik et al dataset was both interesting and a challenge. Several programs were required to accomplish this task including Tika, Grover, DCGAN, and LaTeX. The information was first extracted from the articles listed in the Bik et al paper using Tika. The texts from these papers were then used to train Grover, which then generated text for our 500 falsified papers. The images from these original papers were also extracted and used to generate falsified images using DCGAN. From there, Tika-Dockers was used to generate captions for the falsified scientific images. These generated features for the falsified papers were put into LaTeX which created PDFs for each of the new falsified papers. Finally, using random sampling, the ancillary features that were mentioned in the original Bik et al dataset were also falsified for the falsified papers and included in an updated tsv file.

Extracting the Original Data

In order to extract the text from the 214 papers using Tika Python we needed to automate the download of the 214 pdf files of the papers, which required first installing Chromedriver, Selenium and Tika Python. Installing Selenium and Tika was straight forward, but we noticed that Tika takes 10-20 minutes to start up every time we run code that initializes it, and with some research we realized that this is due to the 60 MB of Tika get downloaded from the server every time its initialized unless you set the TIKA_SERVER_JAR and TIKA_PATH environment variables to the path where Tika was was downloaded on your machine, which we proceeded to do. Automating the download of the pdfs was done using a Python script that retrieves the DOI of each paper from the Bik dataset, where each paper was also given a serial number from 1-214. The code then uses Chrome web browser through Chrome webdriver and Selenium to go to https://dx.doi.org/ and resolve the DOI to get the url of the website that has the published article. The code then grabs the html source of the article website, from which Selenium grabs the download link of the pdf of the paper, downloads it and names it using the serial number it was assigned in the Bik dataset. It was challenging to download the 214 papers automatically in one go as Chromedriver crashed on the first trial after downloading 30 papers, therefore we split the bik dataset with the DOIs into groups of 30 and ran the code one group at a time which required more than one hour to run. The next challenge was finding that many websites use Cloudflare which recognizes Selenium as a robot and blocks it from proceeding with the download. Many attempts were made to work around this including setting a wait time of 10 seconds for Selenium, setting the user agent in the header to match that of the browser on the machine running the code, using BeautifulSoup and using the get command instead of Selenium, and finally using a VPN, all of which were only partially successful in getting the remaining papers automatically and ultimately a few papers were downloaded manually to get all 214 papers. A Python script that uses Tika Python was used to convert the 214 downloaded pdfs into 214 json files which were used as the input for Grover.

Grover Falsification

Grover was an extremely difficult tool for us to use in this assignment. The Github documentation was very unclear and there were many libraries and softwares our team was not familiar with. These included understanding the GPUs/TPUs involved as well as learning how to use Google Colab for this process. There were various bash commands with the Python code that were difficult to interpret and implement. Google drive connection had to be understood and established in order to run the discrimination portion. File paths also had to follow a specific order otherwise the Grover Google Colab file was not able to process the correct input/output. There were also no resources outside of the GitHub documentation for us to reference for help. Restarting the runtime constantly also caused many hindrances along the way. The runtimes were also excessively lengthy adding up to several hours for generation, discrimination, as well as testing the code itself.

Specifically for the generation phase, formatting the input into a jsonl required us to research the file type and then process the Bik dataset into one. Once completed, understanding how Grover would use the Bik dataset to generate 500 articles using the existing data was another challenge. We struggled understanding how to format the code to run multiple iterations and train the model on all 214 articles. Once output was generated, we had to manually filter out 500 articles as 642 were generated from running the Grover model 3 times on all 214 articles. We also had to dig deep into the Grover output itself to understand what was actually the fake text we were looking for as the output contained many more parameters than we had anticipated.

As for the discrimination portion, understanding which model (untrained versus pre-trained, etc.) to use was complicated, as was understanding how to give the model checkpoint files to the discriminator. We also had to pass a very specific format for the input data, which required a python program to create the jsonl input file. The output was returned in a numpy file, which was fortunately easy to process and extract the desired machine data from in order to populate the new tsv file.

Looking closely at the fake text, we were surprised to see how different it was from the original texts inputted. It was very short compared to the papers passed in, often made no sense with many of the words and sentences haphazardly put together, and was not very believable as scientific literature. We think that these papers would be detectable to humans as false media, because the fake texts simply aren't long enough or detailed enough to pass for scientific literature.

Image Generation

This section contained a two-step process that utilized the pdfs generated in task 3 of the report. Task 5 required directories to be created and labeled as <paper name> -images. Python libraries: Glob, os. Path, Fitz, PyMuPDF, and Slugify were used. Pdf files were read using Glob, PyMuPDF obtained the meta-data 'title' info from each paper, Fitz helped scrape images; while converting .jpg to .png, and os.path organized directory paths for Slugify to make the new directories with.

The original plan to employ images obtained from the DOI papers with the DCGAN was insufficient due to technical issues, so ~5k celebrity .jpg files were used as a substitution. Google Colab was the selected IDE to execute the DCGAN generator. DCGAN has a process that uses modeling generator and discriminator tools predefined in Keras Sequential API. The generator aids in creating images that look real, while the discriminator can analyze if an image is real or not. During the training process the generator loops through multiple layers producing slight changes to the original photo with random noise levels until the discriminator can no longer identify the photo as a duplicate, also known as an epoch. In our training sample, 5 epochs were used to produce our fake author faces. Images appear different from the original but are not believable. More training and fine-tuning of the DCGAN generator would be necessary if better results are wanted. There were a few challenges encountered like expired libraries, fixing meta-data title information to generate directory names correctly, and needing to use Google Colab to execute the DCGAN-TensorFlow program since the local machine kernel kept crashing.

Image Captioning

Initial installation of tika-dockers was a bit difficult based on the documentation on GitHub, it was not entirely clear what was required in terms of installation. The next encountered challenge was trying to test the im2textrestdockerfile using a URL that was producing errors. This issue was resolved once an updated URL was provided. The following step was to create URLs for all of the images in order to run them using the following link "http://0.0.0.0:8764/inception/v3/caption/image?url=" which provides a webpage containing information related to the generated caption. The webpage data was scraped and the first caption was saved as the caption for our image. There were no issues that needed to be addressed with the images being of too high quality, as the fake images were actually quite grainy. Postimage.org was used to upload the images and create a url that could be used by tika-dockers.

Unfortunately, the captions do not make sense for our fake scientific articles. This is due to the training model used by DCGAN where the fake images then resulted in faces instead of scientific figures or images. As a result, the captions generated by tika-dockers were descriptors such as "Man holding up a hand in front of a stop sign". Furthermore, because the images were all of faces which looked very similar, many of the captions were repeated. For example "a black and white photo of a black and white cat" was a caption generated for over 50 different images. In the end, it was very difficult to generate convincing captions for a scientific article with the fake image inputs that were utilized.

Falsified Ancillary Features

Most features were randomly chosen from the Bik Dataset (sample distribution). Affiliations and titles were randomly sampled but the affiliation for the fake articles determined the additional features (those extracted from other datasets for assignment one) such as annual growth, education, airquality. Authors were generated using a Python library called Faker. In order to generate fake citations, a list of citations was created using the Bik Dataset. Then, numbers '3' and '4' were replaced by random numbers from a list of numbers. Test code was used to determine if the values were actually unique and did not exist in the original Bik Dataset. DOIs were generated in the same fashion as the citations by sampling the original data and changing numbers '3' and '4' with random numbers.

Final Dataset Observations and Discussion

Thoughts on Final Dataset, Falsified Images, and Grover

While considering how to generate fake data for the final dataset, we realized that certain features relied on other features. For instance, the DOI contained similar values as their corresponding citations. This caused issues in our approach because someone observing the patterns of matching citations and DOI can clearly see that the generated fake values did not correspond. Other approaches should be considered to generate data that better conceals the fact that the data was randomly generated. We were able to correctly assign values for other columns such as the origins country based on the affiliation.

For falsified images, we used face images to train DCGAN, therefore, our images were not scientific graphs/figures. It is quite easy to see that the image is actually a face, although it does appear that the process is incomplete and the image is of grainy quality. The images would be believable as blurry profile pictures, however, they are not believable in terms of images that would be found in scientific articles.

Furthermore, the Grover generated text was not very impressive. The text length is very short, and seems very unlikely as most scientific publications are at least several pages long. We are not experts in the text content subjects, so we are unable to discern whether the text provides valid and interesting information. We wish that there was a way to specify the length of the requested text from Grover. For example, if we could request enough text to fill 10 pages of an article, it would be much more believable as a scientific paper.

Using Associated Ancillary Features from Assignment 1 to Discern Falsified Data

The ancillary features chosen from assignment 1 were able to mask the fact that the papers were fake quite well. Since the data was based on year and country, we were able to correctly assign features according to the country and year a particular paper was published. Also, other features such as 'Retraction', 'Correction', 'No Action', and 'SUM' are dependent on the previous cells. So coming up with a logical way of calculating the values of each column allows for the assignment of values that would be feasible. When generating fake names of authors, considering which country the paper was published to create a name that matches the region's language would make the paper appear more believable.

New Papers Detectable as False Media

Based on Grover, there was an accuracy of 0.263 so it seems unlikely that our papers would be detectable as fake. When looking at the overall papers, including text, images, captions etc., the new papers also seem like they would be detected very quickly. The image quality is very low and the captions are unrelated to the images or the content in the papers themselves.

Ancillary Metadata Features vs. Content Based Techniques

In order to create strong falsified data that appears realistic, it seems both metadata features and content based techniques would be required. However, when it comes to detecting fake data and solving media falsification, metadata provides much more information on where the paper has come from and other important details such as paper origin, author names, last date of edit, software used etc. This would be better over content based techniques as content can still be made to be readable and semi-understandable, whereas fake metadata is very difficult to make real based on our experience with this assignment.

Other Types of Datasets That Could Have Generated Falsified Papers

A dataset of Powerpoint presentations or HTML page sources related to scientific articles are two additional MIME types that could have been used to create falsified papers. The general ideas used in this project would still apply: save the text and associated images to train the fake generators. These tasks could just as easily be done with Powerpoint and HTML files. However, HTML files may be even more useful because Tika could have been easily used to extract content as well as metadata which could have then been used as an example to create fake metadata.

Grover's Ability to Detect Modification on the Original Bik Papers

The Grover discriminator had an accuracy of only 0.263, so it seems that either the discriminator is not doing very well, or the Grover generator is actually doing better than we think. We don't think that the ancillary features from Assignment 1 would have helped to identify the fake media - however if there was a way to check whether the text content matches the title or the author's domain, that could be useful in locating fake papers as most of the papers have random titles that don't correlate with the text subject.

"Backstopping" Required to Generate a Believable Paper Trail

In order to make the scientific literature more believable, certain features could also be falsified in order to give the articles more merit. For example, creating fake web pages on doi.org would make the generated fake data more credible. Additionally, publishing the fake .pdfs online could also support the fake generated information. One can even consider creating a fake website to host the published fake papers. Finally, creating fake social media profiles of the fake authors that match their profiles could be helpful as an additional backstopping feature to make the falsified paper more believable.