Project 4
Title: Deciphering categories & context of handwritten/typed text in images from Illicit & Counterfeit Medicines (ICM) marketplaces
Grad Student Mentor: Timothy Burt (taburt@uh.edu)

1. Motivation

ICM refers to pharmaceuticals that are not genuine. These products can harm both witting and unwitting consumers in a complex causal cycle and infringe on companies' trademarks. This class project is part of a larger project (abbreviated as FIND-M) whose goal is to develop approaches to identify, intervene, and disrupt networks involved in the ICM process.

One of our stakeholders from an unnamed pharmaceutical company has provided us with a sheet of leads, which holds the URLs of websites selling these products and other Personal Identifiable Information (PII) gathered by investigators. We've developed a tool that uses AI & network analysis to cluster groups of related ICM sellers; this project will expand upon it by bringing in new sources of OSINT to the FIND-M pipeline. OSINT stands for Open-Source Intelligence, which refers to any publicly available information.

Method summary

Our method has three stages: the first step is to scrape images/text from the URLs provided. We have already constructed a dataset of over 2,000 HQ images of medicinal products, which we call the Company X dataset. A diagram of the process is shown in **Figure *1***.

Figure 1. Methodology of the data mining process used to build the FIND-M network.

The second step is to analyze the data. This includes preprocessing the images & text, obtaining/processing additional sources of OSINT, and implementing each feature used to link or cluster the leads.

The last step uses network analysis to create a graph representation of the ICM seller community. The nodes in the graph represent each entity from the lead sheet, and the edges represent the number of connections found between the two leads. Each feature is a separate network at this point; to create a single-layer network, parallel edges are merged by summing the weights. A community detection algorithm is performed on the aggregated network to find related ICM sellers

on various platforms/websites, which partitions, or clusters, the network into groups of linked sellers.

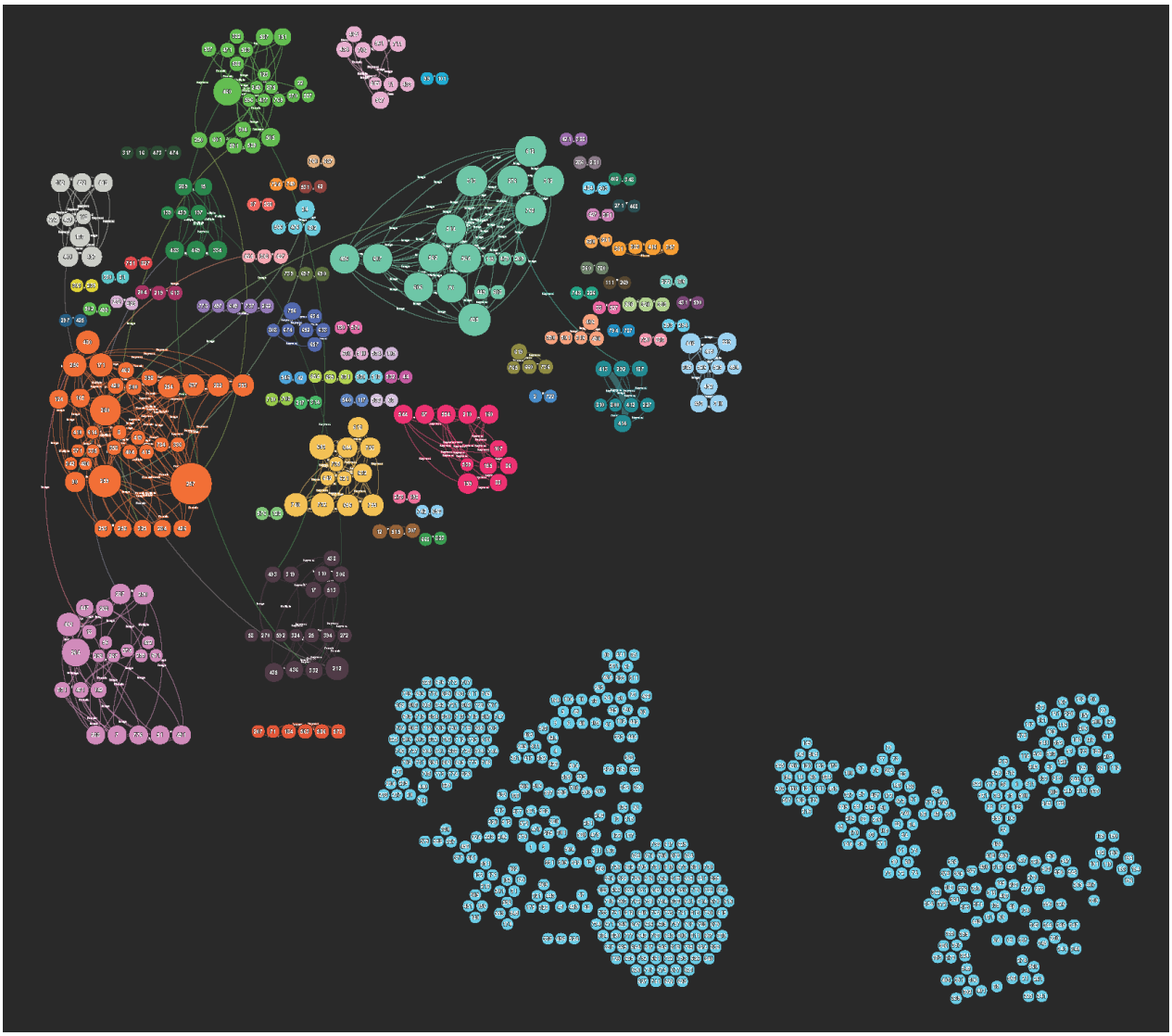

Figure 2: Visualization of the (unabridged) FIND-M network. Clustered nodes are shown with the same color and are positioned in a hierarchy.

Feature construction

Various codes, tools, and packages are used to compute the image & text similarities needed to construct the features, creating pairwise links between entities (each row in the lead sheet). We currently have six features that we hypothesize are good indicators of related sellers: images, phone numbers, emails, keywords, locations, and domain names.

Five features link entities based on similarities that are inherently explainable (a critical need for investigations). A similar image feature is more speculative currently as it lacks the same explainability of the handcrafted features.

Similar image: This feature compares each image in the dataset with every other one by looking for similarities. The package imagededup was used to carry this out with the CNN method. This method loads a Keras-based MobileNet architecture sliced at the last convolutional layer, pre-trained on the ImageNet dataset. GlobalAveragePooling was added at the final convolutional layer, which replaces the fully connected layers used for image classification in MobileNet. The CNN method can find not just exact duplicates but also near duplicates.

Exact phone number/email: There is no ambiguity for these features, and only a small amount of variation (fuzziness) was allowed during the string matching to account for typos.

Similar domain name/keyword: More variation was allowed when comparing these attributes.

Similar address (location): This feature was tedious to implement. It involved standardizing and parsing international addresses, along with a complex record linkage scheme to compute every possible permutation of address and compare them (often using partial information).

Cluster descriptors

In addition to the interactive network, we created a way to briefly visualize the matched features related to each cluster. This was needed to annotate our results and perform an evaluation so we could improve our method. An example is shown in Figure 3.

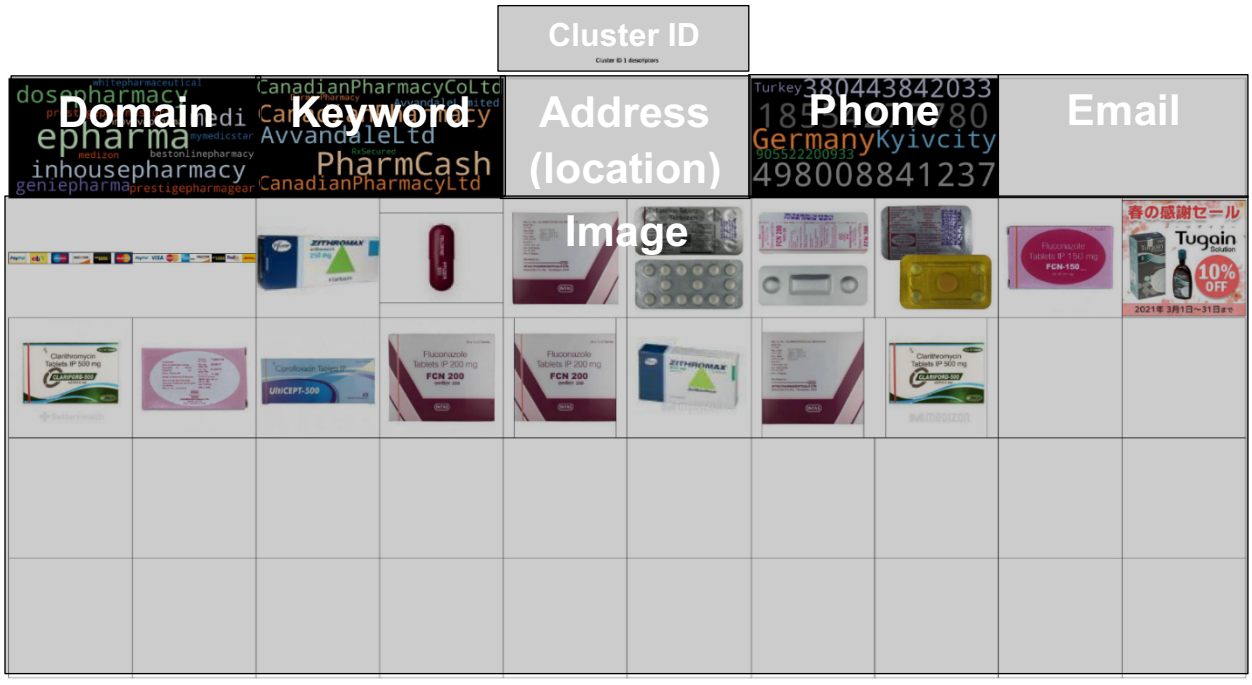

Figure 3: Representative example of the word cloud and image gallery cluster descriptors for a single cluster of ICM sellers, with regions highlighted & labeled for clarity. Top row: Word cloud representations of the matched strings. Lower rows: Image gallery of the similar image matches.

Project description

While you won't be working on steps 1 or 3 exactly, your work will play a pivotal role in the project and will give investigators an indispensable tool in the fight to stop counterfeit medicines from being sold. They are explained here to provide context & motivation for your project.

Your project will fit into step 2; your first task is to perform OCR on a subset of these images. We've already run PGNet with PP-OCR naively on all our images, using the pre-trained model. This is a SOTA algorithm that does not require training or ground truth data. However, this comes at a cost.

Example output from PP-OCR on an image: `['kadcyla' '10omg' 'lnturyoniaik' 'coreth' 'korthantresi' 'thuruimuib' 'onttanven' '10o' 'mg' 'adceeris' '50ng' 'cetris' '5om' 'brenthai' 'edotuin' 'vedotin' 'synpitan' 'forte5luimi' 'lmv' 'sympitan' 'forte' 'lmly' 'enjieksiyoniuk' 'cozets' 'keren' 'ampu' 'enyesyonu' 'conld' 'mpld' 'akos' 'takore']`

A different one: `['app' 'add' 'other' 'contacts' 'nformation' 'my' 'telegram' 'patel' '7580' 'dingtalk' 'patel17580' 'line' 'id' 'patel7580' 'emai' 'vp704029gmail' 'com' 'use' 'app' 'scan' 'code' 'accept' 'card' 'any' 'my']`

How can we bring insights from these jumbled strings? Can we use a pre-trained NLP model to autocorrect these words, join them, and infer the most likely category they fall into (keyword, phone, etc.)?

Some examples of images in our dataset are shown in Figure 4.

Challenges for running image OCR on our dataset

- Different languages: May include English, Chinese, Arabic, and others.
- Handwritten/typeset text: Can be on objects in the image (handwritten), overlayed typed text, or both.
- Handling spaces & special characters: spaces are not typically handled during OCR; the software returns a list of words and is joined afterward. Additionally, some special characters we need may not be in the pre-trained characters.
- Wide range of objects and scenes: mainly medicinal products, but even those vary a bit.

Figure 4: Example images from the Company X dataset which OCR will need to be performed on.

- Joining the output and automatically labeling the text output: Needed to place the correct words/phrases into the right feature category to match the other leads.
- Evaluation of results: Are ground truth annotations needed? Does our model perform well enough for this task?

This project will add new data into our pipeline before the linkage of the features, which may significantly improve the performance of our algorithm.

2. What will the students learn

- The students will get to develop, train, and test SOTA algorithms for image OCR
- The students will learn methods for image analysis & manipulation
- The students will learn various NLP methods & how to implement them in actual use cases
- The students will develop their skills in data science, data wrangling, and entity linkage

Figure 5: Example results from running PGNet using the PP-OCR framework on an image in our dataset, with both handwritten PII and typed medicinal packaging. The text is output on the right, and highlighted boxes are shown on both images around textual areas.

3. Tasks

T.1) Install PP-OCR on your machine (recommend using the pre-built Docker image)
T.2) Read the tutorial about PP-OCR and the PGNet paper (just get a basic understanding of the algorithm). Also, watch the YouTube video, which does almost the same thing we want to do.

T.3) Run pre-trained PGNet on the complete set of images (documenting all parameters and commands used to run it). We highly recommend running it on an NVIDIA GPU using TensorFlow GPU to save time (Linux or Windows boxes only). Compare your results to ours and confirm if they are reproducible.

T.4) Write a Jupyter Notebook to load the PGNet results into a single Pandas DataFrame. Use the image filenames to associate the results with the leads in the lead sheet.

T.5) Try various pre-trained Named Entity Recognition (NEP) codes & methods in the notebook and classify the list of strings from PP-OCR into categories. Start with broad categories (such as PII, medicines, irrelevant) and work your way down to the five handcrafted features we use to link entities (keywords, address, domain, email, phone). Try some NLP text correction techniques here if the results aren't good.

4. Deliverables

ID	Deliverable Description	Type
D.1	FPresentation	Presentation
D.3	The source code for the method	Source Code
D.4	A report describing the implementation of the method as well as evaluation results	Report

5. Prerequisite knowledge

- Jupyter Notebooks

6. Things that you will learn

- Data visualization & manipulation in Python
- Jupyter Notebooks
- Basics of TensorFlow 2.x deep learning framework
- Basics of utilizing the PP-OCR framework [1]
- YouTube video: Rip out Drug Labels using Deep Learning with PaddleOCR & Python [2]
- Basics understanding of PGNet algorithm & how it works [3]
- NLP: pre-trained Named Entity Recognition (NER) models [4], [5] like Stanford NER, spaCy, and NLTK NE_Chunk
- NLP: text correction techniques [6]
- NLP: entity grounding to provide explicit context about what sorts of things it is looking for (common sense) [7]
- NLP reference (don't need to know all of them): code and hierarchy of all SOTA methods with explanations [8]
- How to use Nvidia GPU in Docker to run TensorFlow (this is harder on Windows and in Beta testing at the moment, so Ubuntu is recommended for this project) [9], [10]
- UMAP algorithm for clustering of high-dimensional data [11], [12]

6. References

[1] *PaddlePaddle/PaddleOCR*. PaddlePaddle, 2021. Accessed: Sep. 22, 2021. [Online]. Available: https://github.com/PaddlePaddle/PaddleOCR

[2] Nicholas Renotte, *Rip out Drug Labels using Deep Learning with PaddleOCR & Python*, (Aug. 22, 2021). Accessed: Sep. 22, 2021. [Online Video]. Available: https://www.youtube.com/watch?v=t5xwQguk9XU

[3] P. Wang, C. Zhang, F. Qi, S. Liu, X. Zhang, P. Lyu, J. Han, J. Liu, E. Ding, and G. Shi, "PGNet: Real-time Arbitrarily-Shaped Text Spotting with Point Gathering Network," *arXiv:2104.05458 [cs]*, Apr. 2021, Accessed: Sep. 22, 2021. [Online]. Available: http://arxiv.org/abs/2104.05458

[4] E. Ma, "How does Named Entity Recognition help on Information Extraction in NLP?," *Medium*, Dec. 06, 2018. https://towardsdatascience.com/named-entity-recognition-3fad3f53c91e (accessed Sep. 22, 2021).

[5] M. Terry-Jack, "NLP: Pretrained Named Entity Recognition (NER)," *Medium*, May 03, 2019. https://medium.com/@b.terryjack/nlp-pretrained-named-entity-recognition-7caa5cd28d7b (accessed Sep. 22, 2021).

[6] E. Ma, "Essential text correction process for NLP tasks," *Medium*, Nov. 17, 2018. https://towardsdatascience.com/essential-text-correction-process-for-nlp-tasks-f731a025fcc3 (accessed Sep. 22, 2021).

[7] M. Terry-Jack, "NLP: Entity Grounding," *Medium*, May 02, 2019. https://medium.com/@b.terryjack/nlp-entity-grounding-d89cf0cbbfea (accessed Sep. 22, 2021).

[8] E. Ma, *NLP - Tutorial*. 2021. Accessed: Sep. 23, 2021. [Online]. Available: https://github.com/makcedward/nlp

[9] TensorFlow, "TensorFlow Install Guide: Docker," *TensorFlow*. https://www.tensorflow.org/install/docker (accessed Sep. 23, 2021).

[10] A. Dydychkin, "How to use Nvidia GPU in docker to run TensorFlow," *vicuesoft-techblog*, Jul. 25, 2019. https://medium.com/vicuesoft-techblog/how-to-use-nvidia-gpu-in-docker-to-run-tensorflow-9cf5ee279319 (accessed Sep. 23, 2021).

[11] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *arXiv:1802.03426 [cs, stat]*, Sep. 2020, Accessed: May 04, 2021. [Online]. Available: http://arxiv.org/abs/1802.03426

[12] "Understanding UMAP." https://pair-code.github.io/understanding-umap/ (accessed May 20, 2021).