University of Auckland

Summer Research Project Final Report

Privacy Defense

department of computer science

Supervisor: Joerg Wicker
Student: Chloe Haigh

February 28, 2020

Contents

1 Project Brief

User privacy on the internet is an important and unsolved problem. So far, no sufficient and comprehensive solution has been proposed that helps a user to protect his or her privacy while using the internet. Data are collected and assembled by numerous service providers. Solutions so far focused on the side of the service providers to store encrypted or transformed data that can be still used for analysis. This has a major flaw, as it relies on the service providers to do this. The user has no chance of actively protecting his or her privacy.

In this project, we will evaluate a new approach, empowering the user to take advantage of the same tool the other side has, namely data mining, to produce data which obfuscates the user's profile. Initial work on search engines showed that there is potential to obfuscate the user's profile based on adversarial learning. We will expand the approach and apply it to new domains.

2 Research

2.1 Background

The project was Twitter based, and specifically focused on the promoted tweets that were targeted to a user based on who they follow and how they interact with the website. The goal was to create an account which had a specific interest (such as sport, cooking, gardening etc.) and collect information about the promoted ads that were targeted at the user. Then the aim was to try and confuse Twitter's targeted ads and interact with tweets that were the opposite of the user's main interest and see how the promoted tweets would change to fit with the apparent change in the user's interests. In a past project, my supervisor had done a similar project with Google advertisements and search queries which provided a good base for this project.

2.2 Web Scraping with Selenium WebDriver and BeautifulSoup

Selenium WebDriver is a headless browser, which provides a framework which is designed to automate actions on web applications. You would then write a python file which controls the browser, such as clicking on links, filling in fields and scrolling up and down the page. Selenium was used alongside BeautifulSoup, which is a Python library for pulling data out of HTML and XML files.

My python script would use Selenium to open up the twitter login page, enter a username and password and then scrape the tweets from the home timeline. Actually getting the tweets from the HTML of the page was done with BeautifulSoup. It would then put all the information contained in a tweet into one list element in the order of Display Name, User Name, Tweet Text and Promoted Status. The tweet element would then be added into another list of tweets, and only unique tweets were added to the list.

It would do this by finding the names or id of the elements and then getting the text contained. However this was more difficult than initially thought as the element names on the twitter homepage are all the same (for example, the element id that refers to the username, display name, actual tweet text and any embedded content or attached links are all the same). If there is an emoji somewhere in the tweet text it will also break up the tweet and create another element, which meant you do not know how many different elements are part of one tweet. And as some people tweet media with no text, or just emojis, it was a challenge to represent it by one element. It should be noted I still have not worked out how to parse emojis because they have a different id to the other tweets and have not worked out how to stitch everything together.

Initially, we were under the impression that we would have to use Twitter's own APIs for the project. However, when they collect tweets they do not include promoted tweets which was the whole point of this project, so I had to switch to Selenium. As a whole, Selenium was easy to use and there was lots of documentation online on the different modules. The main faults were more with the way Twitter displays tweets on the homepage than anything else.

2.3 Tweet Classification

After the tweets were collected and put into a list, they were added into a CSV file with different columns representing the different features. The tweets text then needed to be classified into categories so we would be able to see what was being tweeted about. I tried different models and trained them on over one million Google search queries from my supervisor's previous project. The different models I tried will be discussed in the following subsections. All of this was done in Jupyter Notebook as it allowed for easier testing and sectioning the work into parts making it easier to read.

2.3.1 Choosing a Model

The first thing I tried was to follow an online tutorial which took different types of classification models and then got the accuracy of them based on the training data (the Google Search Queries). Because the file was so big, the way the tutorial advised made the program crash the computer multiple times because it tried to vectorise the table and it couldn't allocate enough space (65.6 GiB). So in the end I took a one percent sample and that caused the accuracy to be so low. The results table summarising the accuracy of the models is shown below.

LinearSVC	0.308291
LogisticRegression	0.175754
MultinomialNB	0.074922
RandomForestClassifier	0.095760

Table 1:

2.3.2 SGD Classifier

The accuracy of the models was low so I decided to try and improve on the model. I decided to use an SGD classifier which ended up being the final model I used for classifying the tweets. I've interspersed some misclassified tweets throughout the section for the reader's amusement.

"Spring has sprung."

– @swainson_keith, *classified as "Spreadsheet Software"*

A Stochastic Gradient Descent (SGD) Classifier uses stochastic descent as an optimisation algorithm. It uses the Support Vector Machine as a feature, which is a supervised machine learning model that uses classification algorithms for two-group classification problems.

The SGD Classifier also managed to take the full list of the Search Queries and in the end the accuracy level of the classifier was **0.454298**. The SGD Classifier was built through the Scikit Learn module in Python. It took a 1% sample of the Google search queries and then it trained the classifier on that portion. The classifier was then exported using the pickle model and added into the main python file.

"This no-need-to-knead gluten-free pizza works like a dream."

– @BBCFood, *classified as "Glasgow"*

While some of my test tweets were sorted into accurate categories, it was definitely less accurate when trying to classify the actual tweets. It is also made worse by the fact that each twitter account has a "theme", for example, Sport, so all the twitter accounts they follow are related to the theme. Despite this limitation, the classifier will still put the tweets into unrelated categories. In a sample of 57 tweets (non-promoted), 5 would be sorted into a correct category but it may not be related to the theme (for example, a tweet about rugby in Japan would be sorted into the Japan category even though it is from a rugby account and the theme of the user's account is Sport). However, I did not set the theme to have any effect on the tweet classifier so it is not the fault of the classifier.

2.3.3 Finding Opposite Categories using a Chi-Squared Test

After the tweets were sorted into their categories, I then needed to find the most opposite category as this would be used as a search query in order to try and move the promoted tweets away from the user's interests into the hypothetical opposite of what they like.

The Chi-Squared test compares two variables to see if they are related. The test was available with the SKLearn module for Python.

I passed in the list of queries and for each query it would make a list of the least to most related categories and I would take the first element (least related) and put that into a CSV file with the corresponding category and number. So when I had classified a tweet I would then look up the category and it would return the least correlated category. The accuracy of this method is questionable, for example, the opposite of the category **german cuisine** is **marriage**, and the opposite of **c & c++** is **job**.

2.4 Manipulating Promoted Tweets with Reinforcement Learning

So now that I had the tweet classifier, the next step was to actually try and change the promoted advertisements. There would be two reinforcement learners happening: one trying to change the ads and one trying to keep them the same. The reinforcement learners would have the reward of interacting (by which I mean **liking**, **retweeting** or **replying**) with specified tweets.

The first reinforcement learner, which was supposed to collect promoted tweets was to take the promoted tweets, classify them into categories, get the opposite of the categories and then search up the opposite and interact with the top tweets that came up for the search. Unfortunately it ran into lots of problems:

- In the middle of testing, Twitter would just stop showing ads.
- At times, Twitter would not show ads at all.
- The ads were not targeted to the user's interests (based on what they were interacting with).
- The ads were all New Zealand based despite me changing my "Trending" location.
- It would show a promoted tweet as the second tweet on the homepage but if you keep scrolling, no more promoted tweets would show up.

It meant that often there weren't any promoted tweets to classify. I would have to test it using my own personal account which for some reason would get at least one advertisement every time. On the best day, there was a promoted tweet every time you scrolled down the page. I have not seen it since.

The second reinforcement learner was supposed to interact with tweets that were in line with the user's interests. So it would just go down the tweet timeline and interact with tweets. But because there were so many actual tweets and barely any promoted tweets it became unbalanced.

I have not gotten much further than this but ideally the rewarding should be improved, and the program should loop and keep a record of the tweets which can show how the promoted tweets change over time (if any show up).

3 Conclusion

I enjoyed working in the Computer Science department as a summer research student. It was the first time I had done anything related to text classification or machine learning and it was very interesting. It was also good to be able to learn about Natural Language Processing all the ways it can be used. I felt that I learned a lot and the skills I learned will be beneficial in my future, both at the rest of my time at university and in my future career.

As this was my first time doing research, it was a huge learning experience for me. Despite the project not working out exactly as it should have, I still enjoyed it. It also helped me in deciding to undertake a 380 project next semester and also motivated me to try and go into a field which will use machine learning.