Dark Net Markets: Investigating the Dark Web

W251 Scaling Up! Really Big Data, Spring 2016
Final Project by Marek Sedlacek & Al Byun

1. Introduction

Project Scenario:
Local law enforcement has recently discovered an illicit online marketplace hidden on an anonymous Tor server. Before shutting down the website, officials were able to use a web scraper to download the content for further investigation. The resulting dataset is too large to analyze on their local network. Your Data Science team has volunteered to help the investigation.

Dark Net Markets (DNM) are illicit online marketplaces where users can buy and sell drugs or other illegal material. They also establish communities where users can share knowledge and services through forums and anonymous communication. DNMs are typically hosted on hidden Tor server and use liquid currency, like Bitcoin to further anonymize transactions.

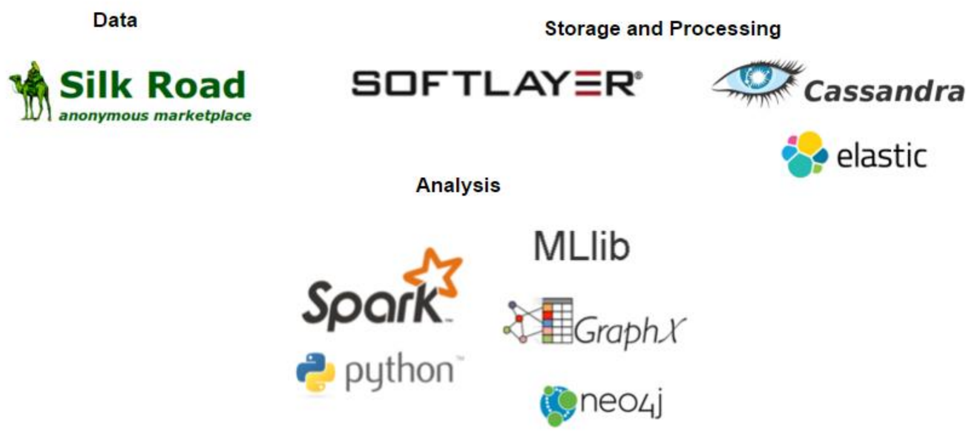
The dataset that we used in our final project was web scraped data from the Black Market Archives (Source: http://www.gwern.net/Black-market%20archives#download). The comprehensive collection is publicly available and includes 89 different DNMs and over 37 related forums. The dataset includes a variety of files, ranging from html, sql, csv, text, and images, for over 1.6 TB of data. Our analysis focused on the two largest and most popular DNMs, Silk Road 1 and its follow on, Silk Road 2. Silk Road 1 operated from January 2011 to October 2013, until it was shut down by law enforcement. Silk Road 2 operated from November 2013 to November 2014, until it was shutdown as well. These two datasets include vendor pages, feedback, forum postings, images, and other information from November 2011 to November 2014.

Here is a summary of our dataset:

Dataset	Volume	Total Num Scraped	File Type
Silk Road 1	11 GB	Profiles: 34,427 Forum: 391,329	HTML, images, text
Silk Road 2	202 GB	Products: 2,156,343 Profiles: 225,306 Forum: 3,909,048	HTML, images, text

Our goal was to provide law enforcement with a deeper understanding of these Dark Net Markets. We set out to provide insight into the popular products for sale, forum topics for discussion, and better understand the network for users.

2. Architecture & Tools



We deployed a cluster of virtual machines on Softlayer to store and analyze the data.

1. Softlayer Virtual Machines

We utilized IBM's Softlayer cloud platform to provision a cluster of three Softlayer virtual machines. Each machine was configured with 4 CPUs, 8192 MB memory, 100 GB local disk, 1000 GB san disk, and CentOS 7 operating system. Additionally, we scaled up to add three more virtual machines in order to decrease processing time when parsing HTML files.

We stored the raw datasets locally onto disk of our main cluster. We then created Python scripts to iterate through the HTML files and parse the data using the BeautifulSoup4 library. BeautifulSoup is a tool to pull text from an HTML document by searching within its tags. Once we had an understanding of the structure of the Silk Road HTML files, we could extract data and organize it as a structured dataframe that would be useful for later analysis.

2. Apache Cassandra

After parsing the HTML files in the dataset using Python, we stored the extracted data into Apache Cassandra tables. Apache Cassandra is an open source distributed database management system designed to accommodate large amounts of data across multiple servers with high availability, fault tolerance, and high performance.

We set up a Keyspace on our main node and set the replication factor to 3. Using CQLSH, we created three different tables within the Keyspace. We then used the Pycassa Python library to load our data into these tables after it was parsed using BeautifulSoup4. As further cleaning and processing of the data was needed (details of which are in the following section), we would import data from Cassandra into Python, make the necessary changes, and re-load the new data back to the tables.

PRODUCTS	PROFILES	TOPICS
CREATE TABLE products(Title_Date text, Date timestamp, Title text, Category text, Price float, Price_Dollar float, Vendor text, Market text, Ships_From text, Ships_To text, PRIMARY KEY (Title_Date)) WITH COMPACT STORAGE;	CREATE TABLE profiles(User_Date text, Username text, User_numposts text, User_type text, User_karma text, Date_registered text, Source text, PRIMARY KEY (User_Date)) WITH COMPACT STORAGE;	CREATE TABLE topics(Author_Date_Responder_Date text, Topic text, Date text, Author text, Author_numposts text, Author_type text, Author_karma text, Response_date text, Responder text, Responder_numposts text, Responder_type text, Responder_karma text, Source text, PRIMARY KEY (Author_Date_Responder_Date)) WITH COMPACT STORAGE;
397,157 rows	50,770 rows	1,394,603 rows

3. Apache Spark

Our main Softlayer cluster had three nodes. They were each configured with Spark version 1.5.0 and Scala version 2.10.4. One node was the main Spark node and the other two were slave nodes. Apache Spark is an open source cluster computing framework, which allows for fault-tolerant, parallel processing of Resilient Distributed Datasets (RDDs) across computing clusters.

We utilized Spark for our data processing and analysis. We attempted to use several HTML parsing packages, including Scala Scraper, to initially parse the HTML data files, however we ran into compatibility and integration issues with our cluster. Therefore, we ended up parsing the HTML files using Python scripts deployed in parallel across multiple nodes.

We did use Spark for interacting with the fully parsed data in Cassandra. We used the Spark Cassandra Connector to connect to our Keyspace and create custom filtered RDDs from our Cassandra tables. We then created Scala scripts to manipulate and analyze these RDDs to answer questions and further investigate the data. The output of these queries were stored as text files and uploaded into Python for creation of final plots, which are presented later in this document.

4. Apache ElasticSearch and Kibana

Once the Cassandra data load and preliminary analysis was complete, we wanted to host the data in a manner that it could be easily accessed and explored by anyone. ElasticSearch is an open-source, distributed real-time search tool that is highly scalable. Kibana is a visualization platform designed to integrate with ElasticSearch.

We created a single index in ElasticSearch for all of our data, and an single type for each table in Cassandra. We utilized the Python package to bulk load data from Cassandra into ElasticSearch, and then set up Kibana to connect to our session. While the tools were limited to simple analytics, it provided us with a means to present interesting findings from our data in an interactive fashion.

5. Apache GraphX

Apache GraphX is a distributed graph processing framework that is built on top of Spark. We used Scala scripts to gather data from our Cassandra tables and create RDDs for vertices (Dark Net Market users) and edges (relationships between users). We then used GraphX to better understand the graph network by analyzing graph analytics like degree and betweenness and by using GraphX's PageRank algorithm.

6. Apache MLlib

Apache MLlib is a Machine Learning framework that is built on top of Spark. It allows for the implementation of many common machine learning and statistical analysis algorithms across Spark's distributed architecture. We used MLlib to extract additional features from text data using natural language processing. Certain features, while not explicitly present in the data, could be inferred by analyzing the text present in the description fields. More details of this process can be found in the following section.

7. Neo4j

Neo4j is a commercial graph database management system. It is a powerful graph analytic tool that stores data into edges, nodes, or attributes. We utilized Neo4j for further network graph analysis and to visualize the network better. We used the Community edition of Neo4j, which is free. However, there is also an Enterprise version available for purchase which allows for scaling up across multiple systems to accommodate larger graph databases.

8. Additional Techniques

a. Softlayer Object Storage

We utilized Softlayer Object Storage, which is based on OpenStack Swift, to store some of our finished data files. It allowed for easy access of files between all of our nodes, as well as our personal machines.

b. Rsync

As learned in one of our assignments, we utilized the Rsync utility to improve our file transfer between nodes. Rsync is a file synchronization and file transfer program that increases transfer reliability and transfer speed.

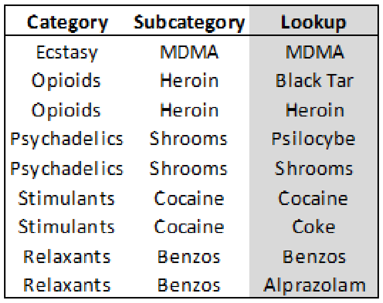
3. Data Preparation and Cleaning

As stated previously, the DarkNet dataset was a large collection of various data, including HTML, csv, and text files. We were able to convert some of the unstructured data to structured data by extracting text that was explicitly tagged in the file. While this worked well for a first pass, this did not produce any data that was only implicit. Also, much of the text was unstandardized and contained gibberish. After parsing the data from the HTML files in it's raw form, we conducted a second pass on the Product data to clean some of the data and create additional variables for each record.

1. Feature Extraction from Text

The initial features we obtained from the HTML files summarized the key details needed to purchase the product (price, shipping locations, vendor) but little classification of the product itself. The product Title was quite descriptive and easy for a person to understand, but the raw text was not suited to machine interpretation. We wrote a script to convert this information into explicit fields in the database, so we could use it in future analysis.

First we needed to classify the products into Categories and Subcategories to compare prices and popularity. Category was one of the original fields we extracted, but the coverage across the population was lacking. Additionally, there didn't seem to be standardization of the categories, so some products would have broad categories while others would be very specific. To classify the products, we began with a simple keyword lookup. Many of the products had the Categories and Subcategories we wanted explicitly listed in the Title. Others could be identified with a list of common synonyms and slang terms (eg. Marijuana = Cannabis, Coke = Cocaine). Finally, if we were unable to categorize the product with our lookup values, we used the original Category to populate the Category and Subcategory fields.



Category	Subcategory	Lookup
Ecstasy	MDMA	MDMA
Opioids	Heroin	Black Tar
Opioids	Heroin	Heroin
Psychadelics	Shrooms	Psilocybe
Psychadelics	Shrooms	Shrooms
Stimulants	Cocaine	Cocaine
Stimulants	Cocaine	Coke
Relaxants	Benzos	Benzos
Relaxants	Benzos	Alprazolam

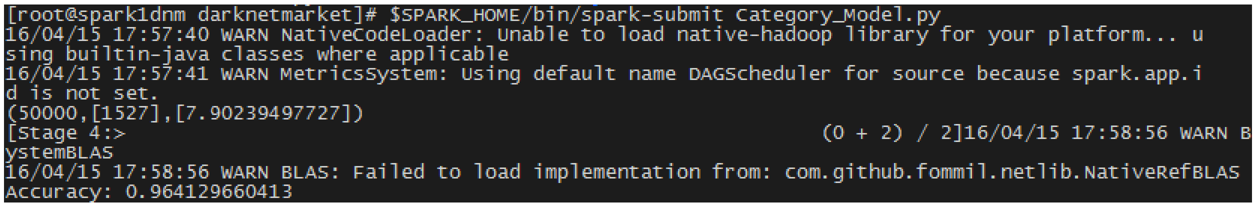
Example of Lookup Table

After categorizing the products, we needed to pull details of the listing weight and count. This information is needed to standardize the prices. Without it we cannot tell if the vendor is selling wholesale or individual, which can cause massive differences in the price. To identify the listing weight, we searched the Title for a list of keywords associated with weight (g, grams, kg, oz, etc.) and returned the number immediately preceding it. For standardization, all weights were converted to grams. If a weight could not be found, it was left blank, as some products would not have weights. To identify the listing count, we first searched the Title for a list of keywords associated with count (X, tabs, capsules, pills, etc.) and returned the number immediately preceding or proceeding it. If no count was found in this manner, we returned the first number in the title that was not equal to the listing weight. While this did lead to some products having inaccurate counts, reviewing a random sample of listings determined it was not substantial. Finally, if no number could be found, the count was simply set to 1.

2. Machine Learning: Category Prediction

Our lookup methodology decreased the missing population substantially. 66% of listings were missing a Category in the raw data, compared with 36% in the cleaned data. However, after cleaning the data we found many products that would be easily classifiable to a human. The lookup list could not discern for minor differences and errors in terms (cocane vs. cocaine), but expanding the list beyond common keywords was a manual and time-intensive process. We determined a machine learning classifier using a bag of words model could provide lift to our categorization rate and give us a better view of the data.

For our training data, we took all of the listings for which we were able to determine a category using the lookup method, as we felt confident in these labels. Due to the high number of categories, we limited the model to only predict whether a listing was one of the top 10 most common categories. Otherwise, we classified it as “Other”. We transformed the title text of each listing into vector of word counts as the feature space for our model. We trained a multinomial naive bayes model on 70% of the data, and tested the predictions on the remaining 30%.



```
[root@spark1dnm darknetmarket]# $SPARK_HOME/bin/spark-submit Category_Model.py
16/04/15 17:57:40 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... u
sing builtin-java classes where applicable
16/04/15 17:57:41 WARN MetricsSystem: Using default name DAGScheduler for source because spark.app.i
d is not set.
(50000,[1527],[7.90239497727])
[Stage 4:>                                                   (0 + 2) / 2]16/04/15 17:58:56 WARN B
ystemBLAS
16/04/15 17:58:56 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
Accuracy: 0.964129660413
```

Spark output from model

The model had an accuracy of 96%. This accuracy is not surprising considering many of the categories were created with a lookup, but it was reassuring. The final step was to take the naive bayes model and predict the categories for all product listings, including those without a category label. As it was unlabeled, we could not test the accuracy on the entire population. However, a manual review of a random sample of 200 unlabeled listings showed the model was producing the correct category 90%. The outcome was not as substantial as the lookup list, but we were able to reduce the missing population by an additional 8%.

Our naive bayes model had a number of weaknesses which, if corrected, could further improve its predictive power. First, our model only classifies listings into the top 10 categories or “Other”. This is due in part to our our limited subject matter expertise in defining the lookup method. While some of the products were easy to classify and group, others were unfamiliar and difficult to identify without extensive research. As a result, while most products fell within a few categories, there ended up being a very high number of categories from the raw data that we were unable to reduce, and the machine learning algorithms struggled with so many classifications. If we had a subject matter expert to help us build our initial category lookup, we could reduce more categories and have a more expansive model.

Secondly, our categories do not encompass non-drug classification. This includes passes to online digital media (Hulu, Netflix), fraud tools (stolen credit cards, skimmers), pornographic material, and other items. Again, if we were able to dedicate more time to building a better initial category lookup we could build a broader list of categories, but this process is manual and time consuming. Finally, in an ideal scenario we would prefer to run multiple iterations of the model on unlabeled data to provide supervised feedback. The initial model would predict labels for listings without categories, and we would manually review a sample of the population and correct the errors in prediction. We would then retrain the model with this additional model and repeat the process until the error rate fell to an acceptable level. As with the other weaknesses, time was a major factor in forgoing this approach.

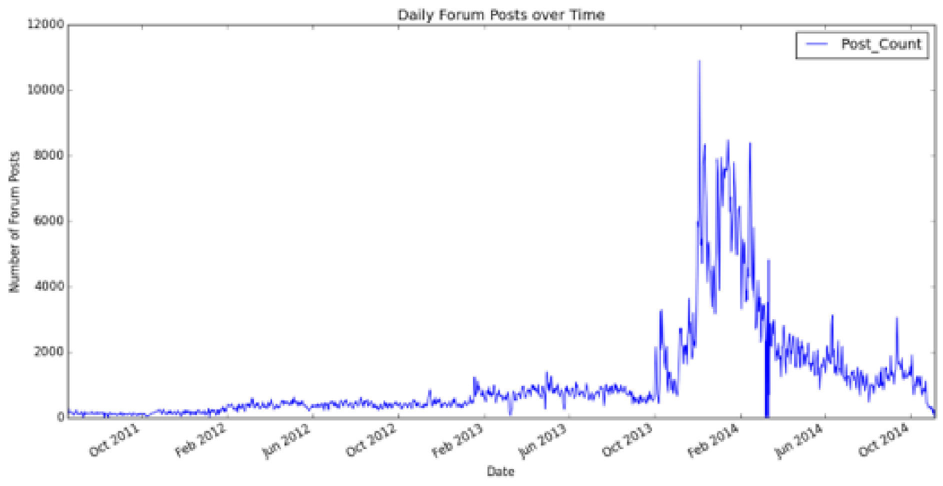
4. Exploratory Analysis

We explored the data using a number of different tools. We used Spark initially to gather a sense of market communication in Silk Road. We then used Kibana to explore the sources and prices of products in the marketplace. Finally, we used GraphX and Neo4j to analyze the interconnectivity of users on the forums.

1. Market Analysis using Spark

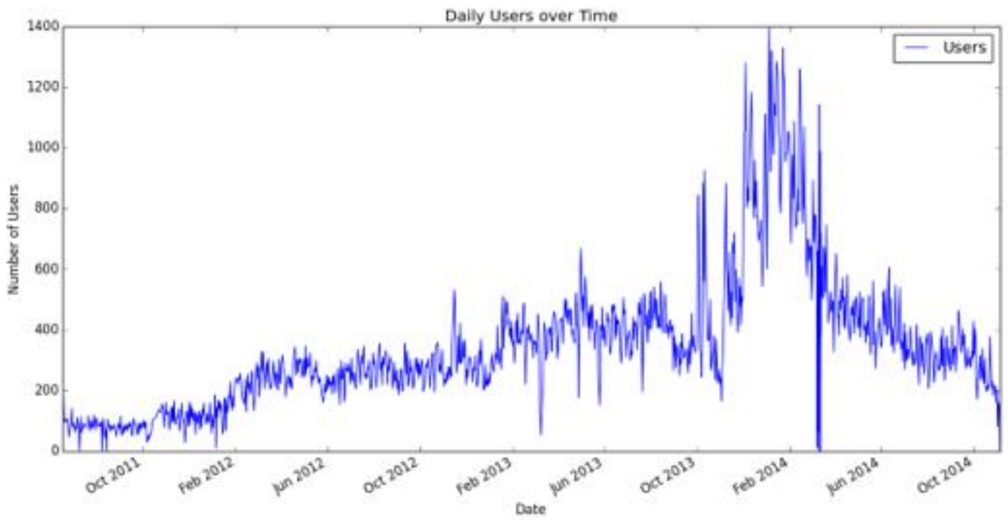
We performed some basic queries in Spark using Scala to better understand the marketplace. In particular, we wanted to know what were the daily activity levels and whether that activity was concentrated or distributed evenly across the users.

How many unique forum posts occured each day?



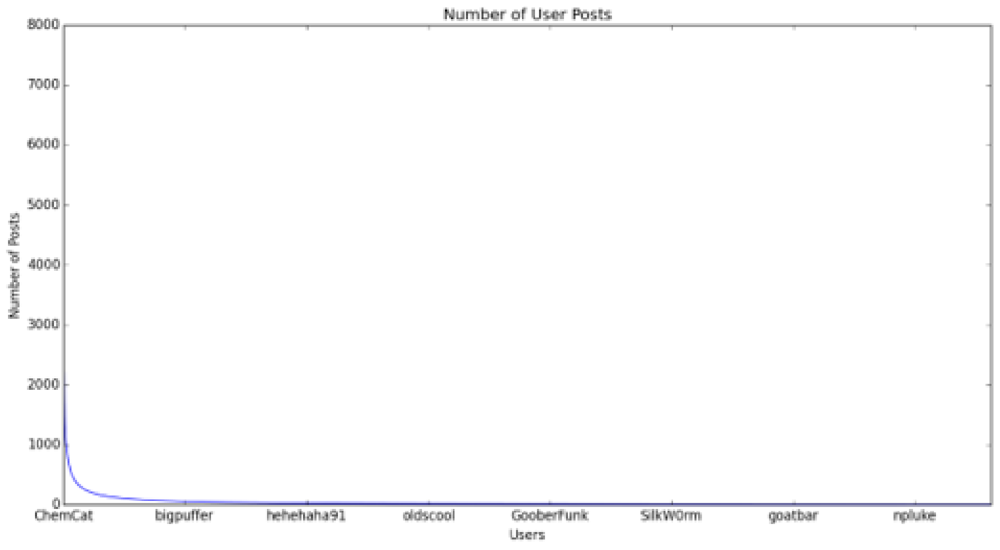
Silk Road 1 operated from January 2011 to October 2013. Silk Road 2 operated from November 2013 to November 2014. Silk Road 2 was enormously popular and rapidly grew to be the largest online Dark Net Marketplace. However in February 2014, Silk Road 2's bitcoin escrow accounts were hacked and upwards of $2.7 million worth of bitcoins from user accounts were stolen. From that point forward, Silk Road 2's popularity declined as users left for similar Dark Net Markets.

How many unique users made forum posts each day?



Silk Road 1 and Silk Road 2 rapidly grew to have at least 52,664 unique user accounts. However, even at its peak, these two Dark Net Markets never exceeded 1500 users per day making posts to their forums.

How many forum posts did each user make?

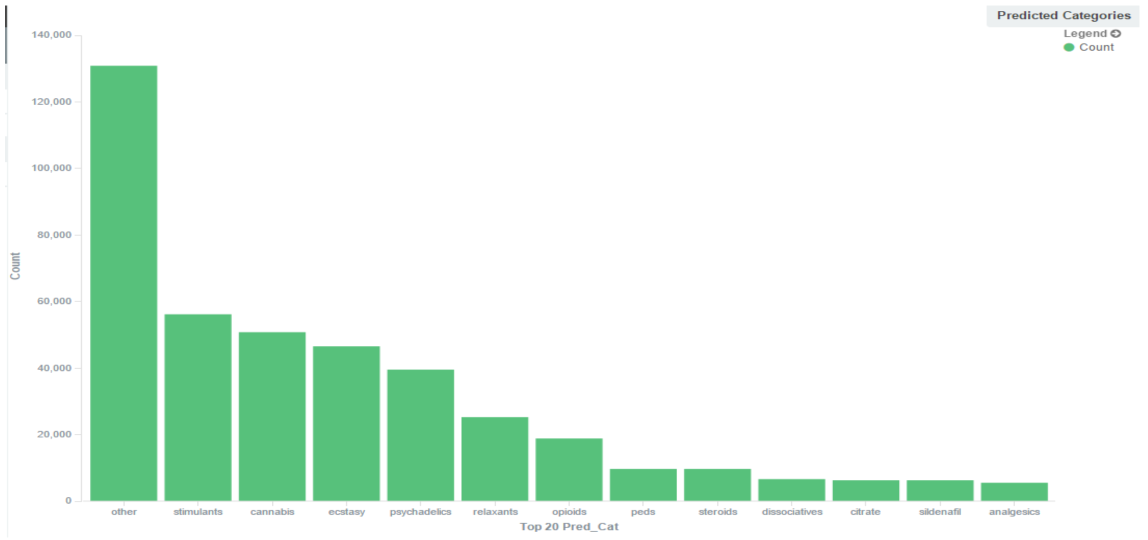


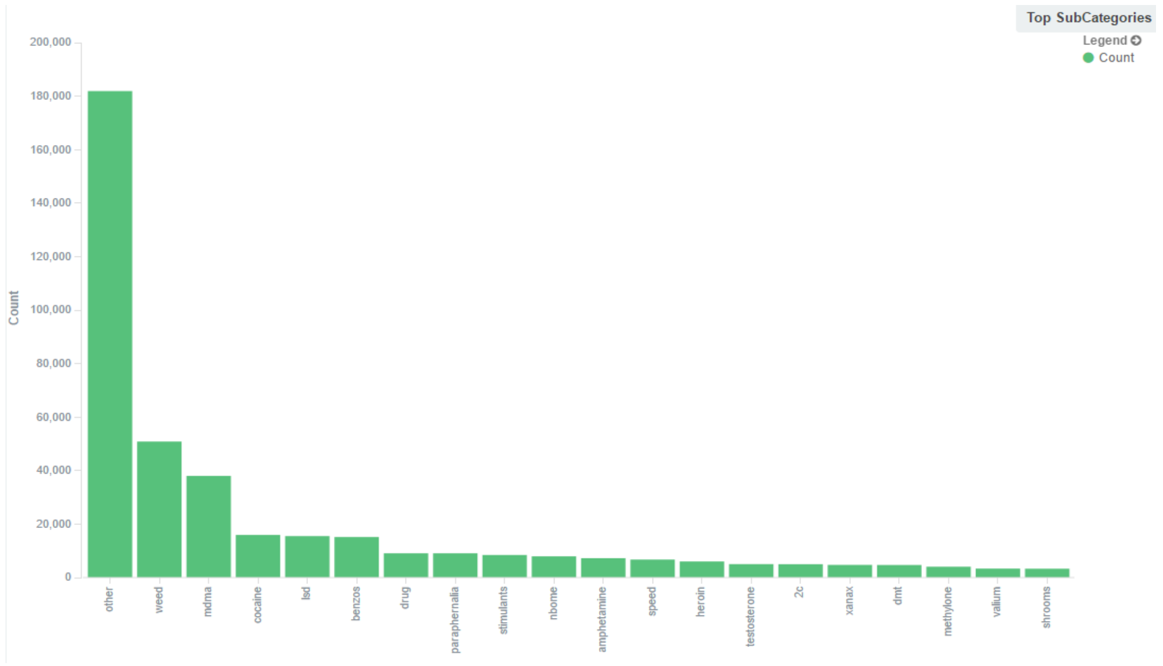
A plot of the number of posts for each user follows a Power Law distribution shape. A majority of the forum posts are made by a smaller portion of the user network.

2. Product Analysis using Kibana

After loading the data into Elasticsearch, we explored the product data using Kibana. We were interested in understanding the product composition of the market, and where those products were coming from. These types of queries could have been performed in Spark reading directly from Cassandra, but Kibana provides a public interface to our data so others can perform their own queries without needing a strong understanding of the underlying systems.

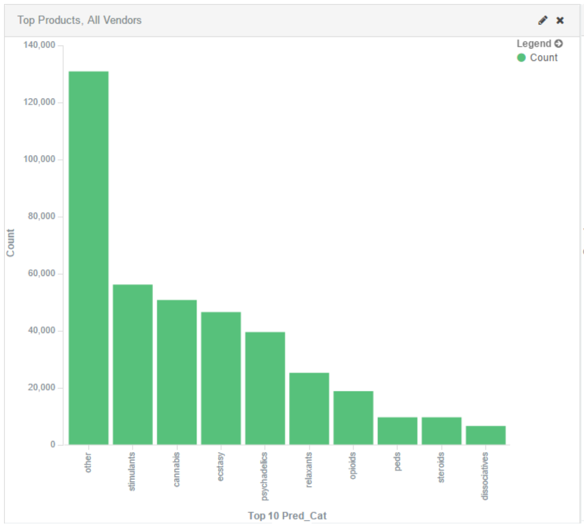
What is the product distribution?

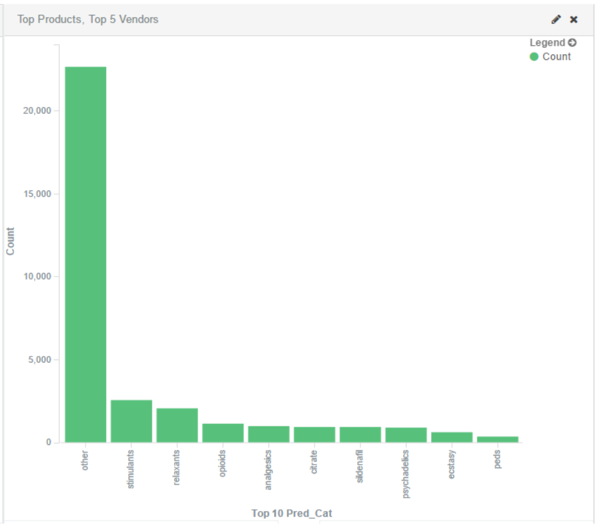




The number one product category was “Other.” Further investigation revealed that while this contained some unclassifiable and small volume drug products, the largest contributor was illegal copies of movies and passes to media websites such as Netflix. Within drug categories, Stimulants were the most common because they contained a wider variety of products (Cocaine, Amphetamines, PVP, and others). Looking only at individual subcategories, Weed and MDMA were more popular than any single stimulant.

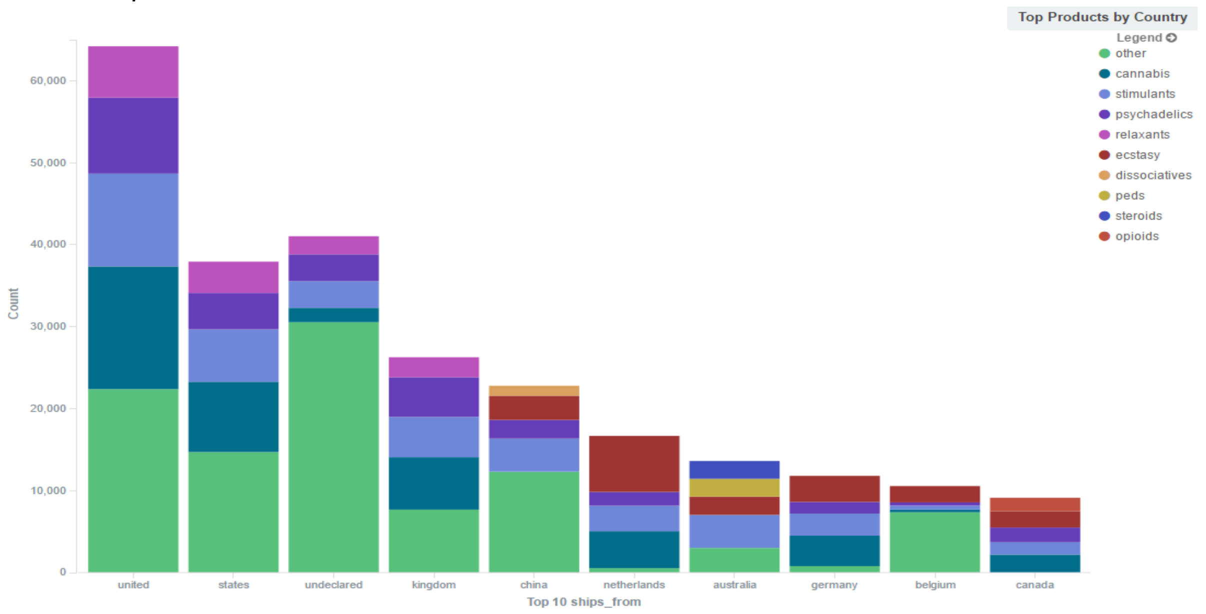
How did top vendors compare to the market?





In our product data there were 2,780 unique vendors, so the top 5 vendors represented less than 0.2% of the total population. But the proportion of products supplied by this small group of sellers is astounding. These vendors were more heavily concentrated in selling non-drug products than the general population. This is intuitive, as it is easier to provide large amounts of digital good compared to physical goods. The top 5 were responsible for 17% of all “Other” offerings, and the tops seller, *profesorhouse,* was alone responsible for 6%. They were also responsible for 8.2% of Relaxants and 4.5% of Stimulant listings. Surprisingly, Marijuana was not a top product for these sellers.

What is the production of various countries?



The chart above shows the top 5 product categories for the top 10 “Ship From” countries. Excluding “Other”, Cannabis is the most common product across the board, with most countries shipping at least some amount and many shipping it most. The Netherlands stands out as a leader in Ecstasy production, owning 15% of all listings. Australia is the most common provider of Steroids and Performance Enhancing Drugs (PEDs), with 22% of the listings.

3. Network Analysis using GraphX

How many unique users (nodes/vertices) are there?

```
val numusers = graph.numVertices
```

- numusers: Long = 52664

How many relationships are there between users?

```
val numrelationships = graph.numEdges
```

- numrelationships: Long = 562116

What are the most frequent/strongest relationships?

```
graph.triplets.sortBy(_.attr, ascending=false).map(triplet => triplet.attr.toString + " from " + triplet.srcAttr +
" to " + triplet.dstAttr + ".").collect.take(10).foreach(println)
```

- 162 from giancarlo to Merde222.
- 146 from ScrewsLoose to Merde222.

- 140 from mary666 to ChemCat.
- 133 from mary666 to CaptainWhiteBeard.
- 122 from murderface2012 to Merde222.
- 116 from ChemCat to mary666.
- 114 from calcium345 to CaptainWhiteBeard.
- 114 from doctorwhat to ChemCat.
- 113 from murderface2012 to ACE.
- 100 from murderface2012 to smity1020.

What are the least frequent/weakest relationships?

```
graph.triplets.sortBy(_.attr).map(triplet => triplet.attr.toString + " from " + triplet.srcAttr + " to " +
triplet.dstAttr + ".").take(10).foreach(println)
```

- 1 from Holland_SR to frogwithADD.
- 1 from fotwentee to Treignrex.
- 1 from fotwentee to one2bcurious.
- 1 from fotwentee to northsouth.
- 1 from robbiefowler23 to googleyed1.
- 1 from VANQUISH4777 to thecatisback.
- 1 from VANQUISH4777 to SandStorm.
- 1 from VANQUISH4777 to TheDemiGod.
- 1 from VANQUISH4777 to SourDiesel.
- 1 from VANQUISH4777 to chakalaka.

What user has the most in degrees or unique users into it?

```
graph.inDegrees.join(vertices).sortBy(_._2._1, ascending=false).take(10).foreach(println)
```

- (546159859,(1726,ChemCat))
- (-1477396411,(917,DoctorClu))
- (254234108,(805,murderface2012))
- (208702416,(799,Limetless))
- (-812369126,(760,Higuel))
- (159345784,(719,giancarlo))
- (-1221072419,(708,CaptainWhiteBeard))
- (820108162,(701,HonoluluExpress))
- (-127955699,(682,moonbear))
- (-187502943,(597,mary666))

What user has the most out degrees or unique users out of it?

```
graph.outDegrees.join(vertices).sortBy(_._2._1, ascending=false).take(10).foreach(println)
```

- (546159859,(2183,ChemCat))
- (209060421,(1966,Tang))
- (208702416,(1192,Limetless))
- (-319647961,(1189,scout))
- (2092013915,(1084,TheSlyFox))
- (1386904961,(986,fallingsnow))
- (-1477396411,(938,DoctorClu))
- (-227248955,(932,cryngie))
- (1998156406,(865,Libertas))
- (254234108,(824,murderface2012))

//**Page Rank**
val ranks = graph.pageRank(0.0001).vertices

Which users are the most important?

val ranksUsers =
ranks.join(vertices).sortBy(_._2._1,ascending=false).map(_._2._2).take(10).foreach(println)

- ChemCat
- Limetless
- DoctorClu
- HonoluluExpress
- murderface2012
- Higniguel
- CaptainWhiteBeard
- giancarlo
- schnitzel_karl
- mito

What are the Top 10 most popular topics for the Top 5 important users?

val topics_user = topics_rdd.filter(row => row(0) == "ChemCat").map(row => (row(row.length-1).toInt,row(1))).sortByKey(false).take(10).foreach(println)

ChemCat
- (484,Re: Official Spare Coins Thread)
- (309,Re: Official Spare Coins Thread READ OP BEFORE YOU REQUEST!!)
- (304,Re: **Spam to 50 &amp; Get out of Noobville**)
- (264,Re: READ THE 1ST PAGE!!!Official Spare Coins Thread READ OP BEFORE YOU REQUEST!!)
- (209,Re: Official Spare Coins Thread)
- (194,Re: Are you Paralyzed by PGP? Fear no more! Join PGP Club :)
- (119,Re: The Green Camel Night Club)
- (70,Re: Newbie PGP Club)
- (60,Re: +++KARMA)

- (53,Re: NEWBIES: Guide for learning PGP)

Limetless

- (111,Re: MEPHEDRONE VENDORS!)
- (50,Re: READ FUCKING FIRST POST! YOUR NAME MUST BE IN IT - The Verbal Diarrhea Thread)
- (37,Re: Chat more shit)
- (17,Re: Limetless reviews)
- (12,Re: + KARMA)
- (10,Re: What music are you listening to right now?)
- (7,Re: Last one to reply wins!)
- (5,Re: Anyone recieve from SuperTrips?)
- (5,Re: Get Vouched or Get Fucked - The Verbal Diarrhea Thread)
- (5,Re: My account was hacked AGAIN by a MOD I suspect!!!!)

DoctorClu

- (67,Re: Vendor Verification for Round Table Access)
- (61,Re: Official Spare Coins Thread)
- (53,Re: +++KARMA)
- (34,Re: The Green Camel Night Club)
- (31,Re: Contingency Plan)
- (25,Re: X)
- (21,Re: StExo was behind the attack. Defcon did his best to defend the place.)
- (14,Re: Buyer Blacklist)
- (14,Re: Full Disclosure)
- (14,Re: Inigo)

HonoluluExpress

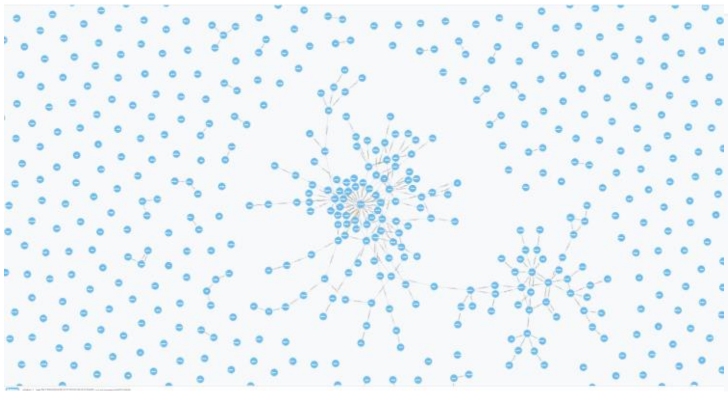
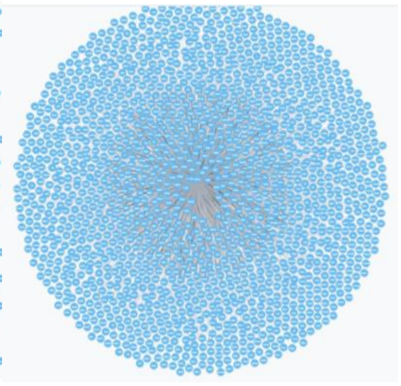
- (88,Re: +++KARMA)
- (48,Re: Official Spare Coins Thread READ OP BEFORE YOU REQUEST!!)
- (45,Re: Official Spare Coins Thread)
- (35,Re: The Green Camel Night Club)
- (23,Re: I'm Here To Help The Newbies! FAQ Now Included In The OP!)
- (20,Re: Official Help The Newbies Thread! FAQ Is Included In The OP! Updated 1/30/2014)
- (19,Re: I'm Here To Help The Newbies!)
- (14,Re: Unofficial List Of Vendors To FE With! Updated Frequently!)
- (12,Re: Official Spare Coins Thread READ OP BEFORE YOU REQUEST!!)
- (12,Re: Relaunch Timeline + Important Reminders UPDATE 2)

murderface2012

- (596,Re: The Scurvy Crew - Reviews and AWESOMENES Home of the finest hash)
- (578,Re: ~OFFICIAL SilkRoad 2.0 Cocaine Thread -- First post modified with vendors~)
- (343,Re: Official Spare Coins Thread)
- (233,Re: Official Spare Coins Thread READ OP BEFORE YOU REQUEST!!)
- (205,Re: OFFICIAL SilkRoad 2.0 Cocaine Thread)
- (164,Re: Chemicals_Spain review thread)
- (156,Re: +++KARMA)
- (155,Re: READ THE 1ST PAGE!!!Official Spare Coins Thread READ OP BEFORE YOU REQUEST!!)
- (132,Re: The Green Camel Night Club)
- (95,Re: ~OFFICIAL SilkRoad 2.0 Cocaine Thread -- No current vendors listed)

4. Network Visualization using Neo4j

We visualized the DNM network using Neo4j. The image on the left is a screenshot of a small portion of the network. It shows the typical formation of one popular user in the center and multiple individual users surrounding that user. The image on the right is a screenshot of the most popular user's, ChemCat, isolated network. These visualizations show that a majority of the users have very small personal networks. These users likely are attempting to maintain a low profile or are not as active. The few popular users maintain large personal networks.

5. Conclusion & Future Work

Our project set out to utilize data science techniques analyze of Dark Net Markets (DNM) to assist law enforcement. We focused our efforts on the most popular DNM, the original Silk Road 1 and its successor, Silk Road 2. Our source, the Black Market Archives (Source: http://www.gwern.net/Black-market%20archives#download), scraped the data from this online marketplaces and forums. It included over 4 million HTML, image, and other files, which was over 200 GB in size.

We downloaded this data onto multiple Virtual Machines, parsed the HTML using Python scripts with the BeautifulSoup library, loaded the parsed data into a Cassandra database, and finally analyzed the data using Spark and its family of libraries, as well as other analytic tools, like Neo4j, Elasticsearch, and Kibana. As a final deliverable to our law enforcement customers, we provided them with a package of powerful tools including, Python and Scala scripts, a trained Machine Learning model for product analysis, a Network graph database with visualizations, and a Kibana search User Interface. With these tools, our customers can better understand the top users on the Dark Net Market forums, general forum behavior and trends, and product behavior and trends. These tools also allow our customers to easily search and discover their own key insights, as well.

In future work, we can augment to this project with additional functionality and tools to help gain better insights to this data for our customers. We would like to add other Dark Net Markets to compare trends across these illicit industries. We would also like to set up our own crawlers, using tools like Apache Nutch, to collect new data and integrate it with Cassandra and Elasticsearch/Kibana to allow for real-time information and analysis. We would also like to better integrate our product and forum data through the analysis of vendor user profiles and create improved Machine Learning models to analyze and predict behavior.

6. Appendix: Setup & Code

1. Softlayer Virtual Machines:

Main Spark Cluster:

VM name	Public IP	Private IP	root password
spark1dnm	158.85.217.74	10.122.116.144	T3jwTq62
spark2dnm	198.23.108.214	10.91.48.126	Uwjru2LC
spark3dnm	198.23.108.220	10.91.48.91	D2xhdk2T

spark1dnm is the master Spark node and spark2dnm and spark3dnm are the slaves.
On each machine, there is a 1 TB san disk that is mounted to a folder called /sandisk1 (or /sandisk2, /sandisk3, depending on the VM).

“cd /sandisk1/darknetmarket” on spark1dnm to access the unpackaged Silkroad1 and Silkroad2 data.

Additional VMs for Analysis:

VM name	Public IP	Private IP	root password
dnm1	50.23.91.3	10.54.254.241	AkZs5unH
dnm2	50.23.91.6	10.54.254.197	Mn2srQ5v
dnm3	50.23.91.11	10.54.254.198	Ha9m77E4

2. Github Repo:

Our code is located here:
https://github.com/Marek-Sedlacek/W251---DarkNetWeb