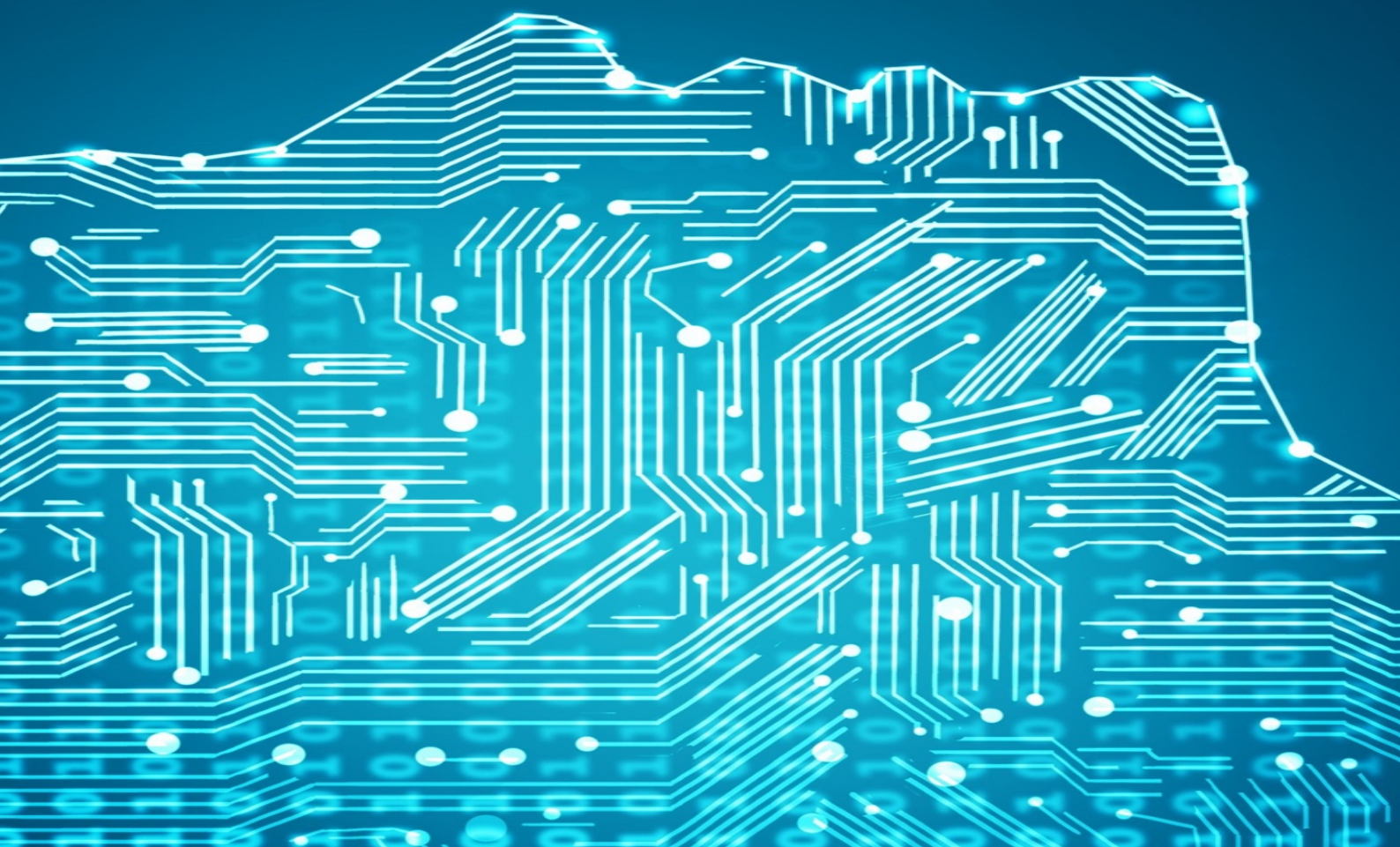
OilX – Refinery Event Monitoring System (Report)

OILX

London Business School

Table of Contents



Abstract

OilX enables traders and analysts to gain a comprehensive view of the global oil markets, and as a result, make better data-driven decisions. This paper addresses a solution that reinforces OilX's value proposition by providing and processing real-time data about refinery events around the world. We first scraped data from Twitter and Thomson Reuters and then built an algorithm that is capable of identifying and clustering similar events. The algorithm was trained on 35 000 headlines & 7 000 Tweets, and it yielded a 77.5% accuracy on a random sample of 165 headlines & 35 Tweets. It was able to process 100 Tweets in 64 seconds on a personal computer with an unmodified CPU. Tweets usually conveyed information faster, while headlines were of better quality. Further improvements involve making smart suggestions (analyzing tense, sentiment, the trustworthiness of sources), filtering out other types of refineries (e.g., gold, sugar), and predicting the impact on refinery capacity.

Introduction

OilX has developed a digital twin of the global crude oil supply chain by combining traditional data with alternative data sets, including cargo tracking and satellite observations. Currently, OilX is looking to develop a system to monitor adverse events that impact oil supply in refineries. To overcome this problem, we developed an algorithm capable of gathering, cleaning, and processing data related to worldwide refinery installations. This data is then used to extract information about the location, ownership, and type of event.

Produced Solution Outline

Our solution consists of five sequential phases – data gathering, cleaning, extraction, identification, and clustering- to produce the desired output of information. **Figure 1** provides a holistic overview of our solution.

Figure 1: Produced solution outline

Data gathering

The first step consists in gathering Tweets from Twitter and headlines from Thomson Reuters. We created a scraper using the Python web scraping library Selenium in order to collect relevant information only. This scraper returns Tweets containing all combinations of refineries-related keywords (e.g, refinery, refineries, etc.) and events-related keywords (e.g., fire, attack, strike). Given the unstructured nature of the data, we used the database management software MongoDB. The use of MongoDB is particularly convenient because it allows us to use all the Twitter API's features including natively isolating hashtags, to which we apply different text-matching methods. **Figure 2** provides an overview of a Tweet's raw output on MongoDB.

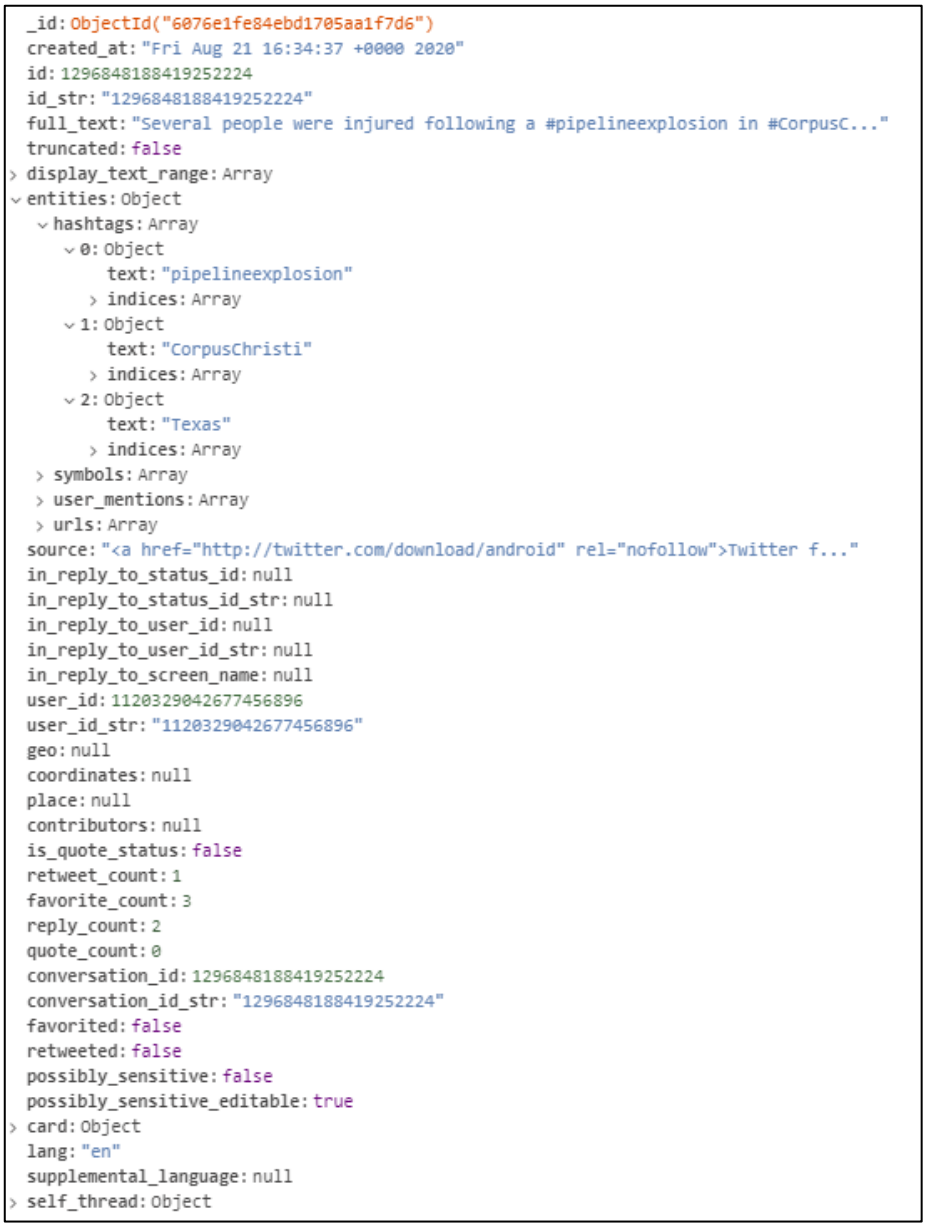


Figure 2: Scraper's output on MongoDB

Data preparation

Our model uses data about worldwide refineries provided by OilX (refinery name, city, coordinates, owners) to match Tweets and headlines to specific refineries. As refineries and cities might be misspelled or shortened in Tweets /headlines, we generated multiple variations of these names (e.g., Los Angeles could be #LosAngeles without spaces). This step is necessary to achieve accurate string matching but only needs to be run when the GeoAssets file or the Owners file is altered. **Figure 3** indicates the data preparation steps.

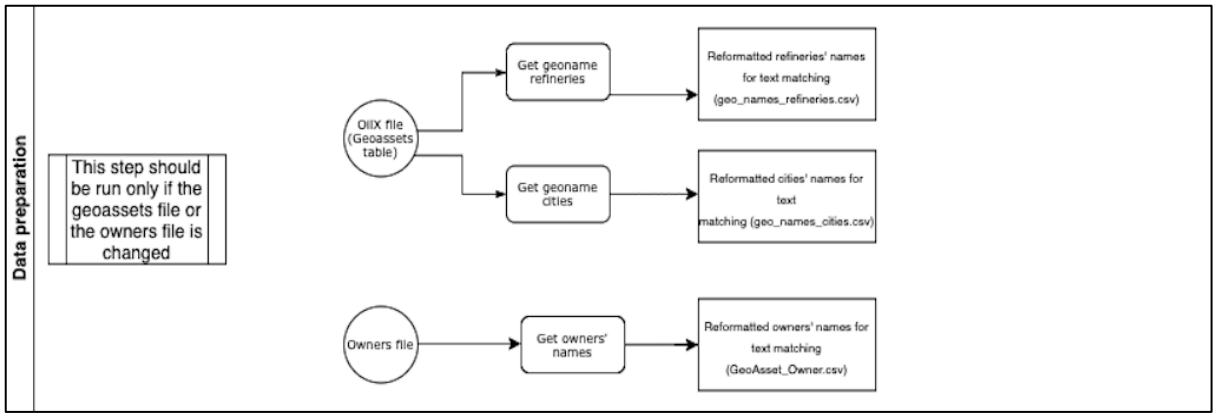


Figure 3 : Data preparation

Extraction

We use text matching in MongoDB to extract reformatted names (saved in the previously-generated files) from Tweets and headlines. We append matched refinery names, city names and owner names to the corresponding Tweets and headlines in MongoDB, and use the python library spaCy with three NLP models to extract GPEs (Geopolitical Entities) and NORPs (Nationalities Or Religious or Political groups). To maximize the number of matches across different languages, we use case and diacritic insensitive text matching. We chose to use MongoDB for its ability to perform these operations quickly and efficiently. **Figure 4** provides an overview of the extraction's steps.

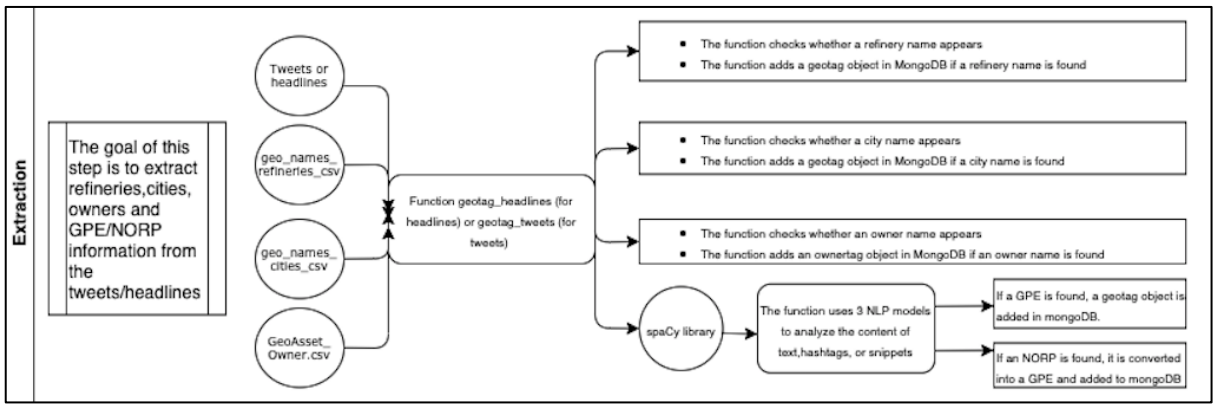


Figure 4 : Extraction

Refineries matching

During this process, the algorithm identifies the refinery to which the Tweet/headline refers. To ensure string matching that is consistent with OilX's database, the algorithm considers the date at which the Tweet was

posted, thus ensuring that the matched refinery was indeed open when the Tweet was published.

We harnessed a four-pronged approach to leverage all information extracted from Tweets and headlines in the following order :

- Text matching by
    1. Refinery name (refinery geotag)
    2. City name (city geotag)
    3. Owner name (ownertag)

- Use of the spaCy library to extract
    4. Geopolitical Entities (GPEs)
    5. NORPs, then converted in GPEs

The use of GPEs requires the deployment of a pre-trained NLP algorithm that will find any references to locations in the text and classify them according to their entity (city, country, or other tags). Moreover, we use the Nominatim API and the geopy library to convert the GPE into a bounding box. The algorithm then analyses all refineries present within this bounding box, according to the coordinates provided in OilX's file. **Figure 5** displays a bounding box and its corresponding refineries.

Figure 5 : Bounding box with Nominatim API (unique match)

Using this information, if a unique match is found, the program will store the name of the refinery in MongoDB and assume 100% certainty. If more than 1 match is found, the program will suggest all possible locations and offer a level of accuracy proportional to the number of refineries available in that location (1/n). This could be done by checking the number of refineries in the mentioned location or the number of refineries that belong to the said owner.

Figure 6 shows a case in which the algorithm would allocate 33% probability to each refinery.

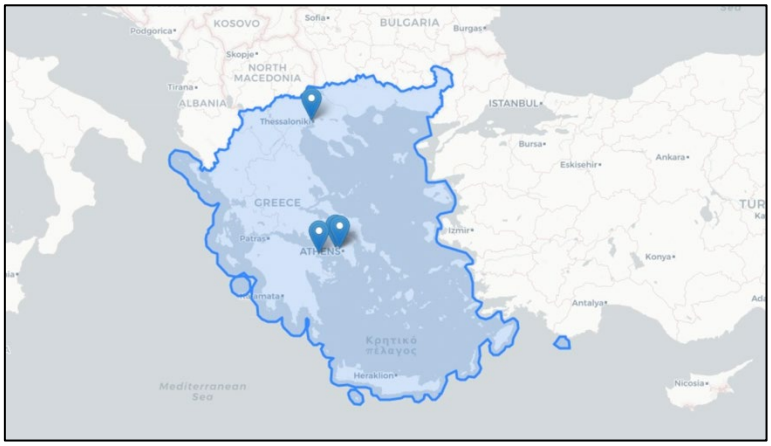


Figure 6: Bounding box with Nominatim API (multiple matches)

Figure 7 summarizes the steps taken by the algorithm to perform refinery matching.

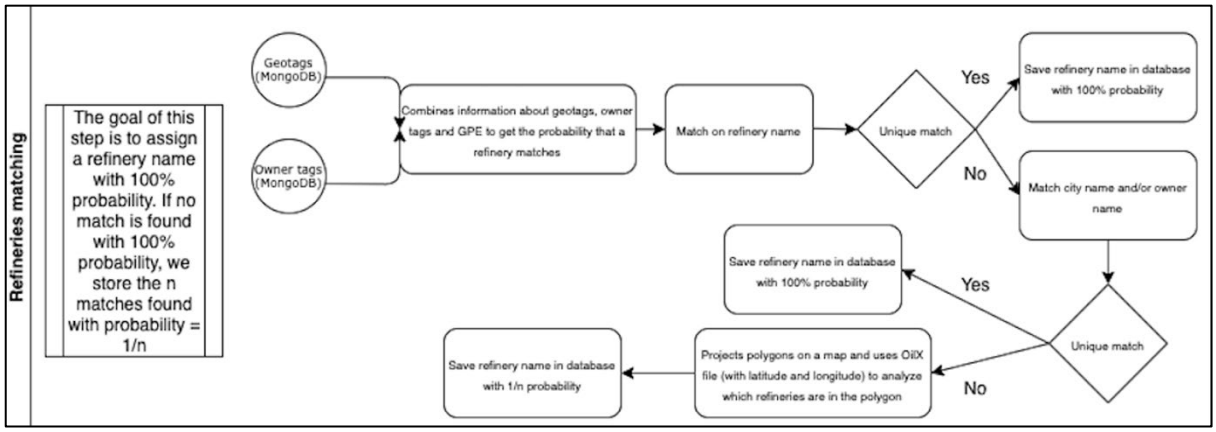


Figure 7: Refineries matching

Events matching

The algorithm once more performs string matching on the raw data and appends the identified event to a list in MongoDB. It is important to note that one Tweet/headline can refer to more than one event type. For example, there could be both an explosion and fire at the same time. This classification will be further used in the clustering section to perform the grouping of similar Tweets/headlines that relate to the same event. **Figure 8** summarizes the event matching process.

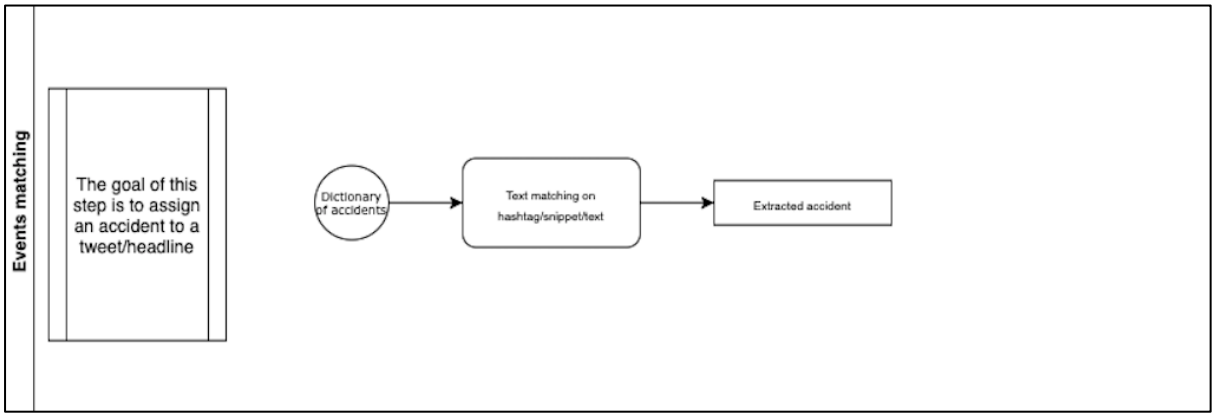


Figure 8: Events matching

Clustering

After classifying the Tweets/headlines by location and event, we cluster them in order to get a single complete output for each event. The goal of this step is twofold. First, it allows to combine information among Tweets/headlines within the same cluster. Secondly, it summarizes all information into a single result in order to have a clear and simple output for traders.

The clustering process takes time, location, and event under consideration.

First, Tweets and headlines are grouped together based on country name and creation time. We chose country names since this is the least granular piece of information, ensuring the highest likelihood that the Tweet contains this information as opposed to refinery name or city name. Moreover, as an assumption, we have defined that it is unlikely to see two separate accidents in the same time frame (1 week) in the same country. As a result, the algorithm generates an adjacency matrix of size (n,n) for each country, with n being the number of Tweets/headlines within the same country. The adjacency matrix shown in **Figure 10** will display 1 if two Tweets /headlines are <1 week apart, and 0 if they are >1 week apart. The algorithm then forms a first layer of time-based clusters (composed of ones) in each country

	12	13	14	15	16	17
11	0	0	0	0	0	0
12	1	0	0	0	0	0
13	0	1	1	1	1	1
14	0	1	1	1	1	1
15	0	1	1	1	1	1
16	0	1	1	1	1	1
17	0	1	1	1	1	1
18	0	0	0	0	0	0
19	0	0	0	0	0	0
20	0	0	0	0	0	0

Figure 10: Time adjacency matrix

In order to make clustering even more consistent with reality, we added a layer of event clustering. First, the algorithm calculated the correlation among all events within our dataset of 35 000 headlines and 7 000 Tweets. **Figure 10** shows a part of the resulting correlation matrix as an example.

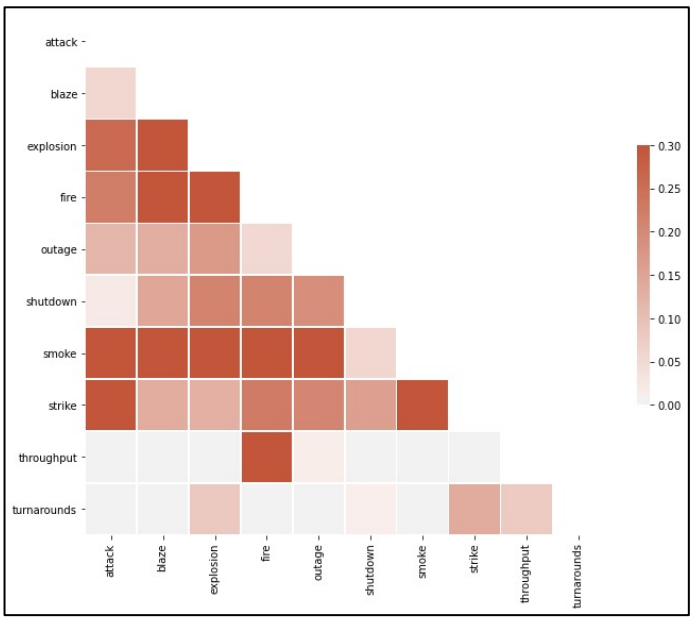


Figure 10: Events correlation matrix

The correlation among events allows the algorithm to put Tweets/headlines in different clusters even though they belong to the same country and happened the same week. If two Tweets/headlines are close in time but mention very uncorrelated events (ρ <0.4), the algorithm will be able to recognize two different events happening and will put these Tweets/headlines into different clusters through an adjacency matrix.

For example, since we know that strike is almost uncorrelated with throughput, the algorithm can separate two Tweets mentioning events that happened in the same country during the same week if one is about throughput and the other is about strike. If each of the 2 Tweets contains many events, the algorithm will calculate correlations between all pairs of events. Note, however that the algorithm imposes Tweets/headlines to be at least two days apart in order to be in separate clusters. [1]

Finally, we superpose the two adjacency matrices (one for time and one for event) to develop unique clusters containing all information relevant to specific events. Furthermore, the clusters will diminish geolocation uncertainty due to a higher volume of Tweets being associated with the unique event(s). **Figure 11** summarizes the clustering process.

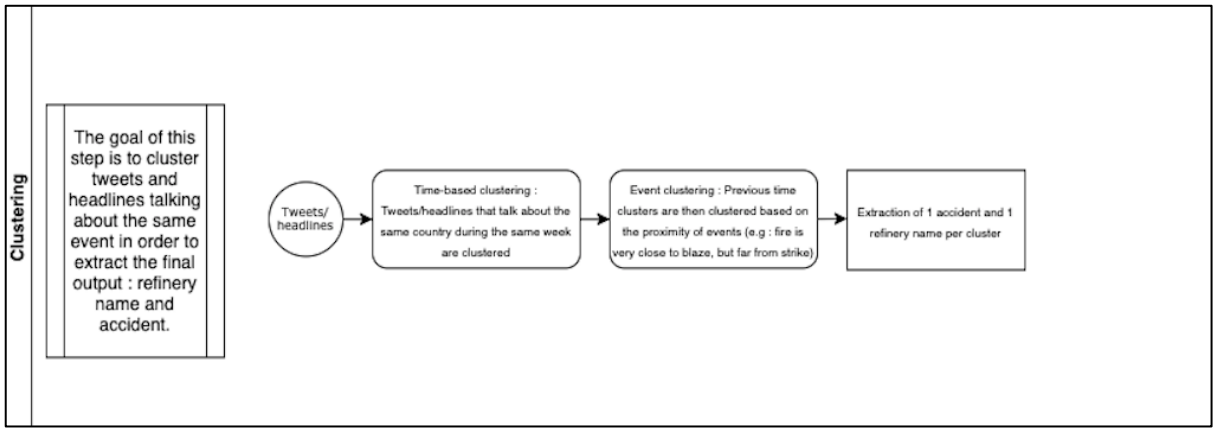


Figure 11: Clustering

Results

We selected a random sample of 35 Tweets and 165 headlines in order to estimate the algorithm's speed and accuracy. It was able to process 100 Tweets in 64 seconds on a personal computer with unmodified CPU, and it yielded an accuracy of 77.5%.

False-positive errors (18.5%) were mainly due to very broad Tweets for which the algorithm could not assign a refinery with 100% probability. This could be highly reduced by simply filtering on refineries found with 100% probability. False-positive errors were also partly due to the algorithm not being able to recognize among negative and positive sentences (e.g., "not to strike" vs. "to strike").

[1] This assumption can simply be changed in the parameters provided with the report.

On the other hand, false-negative errors (4%) stemmed from the algorithm not being able to recognize a refinery name that was not in the file provided by OilX. Although these errors only represented 4%, they could be easily solved by updating the GeoAssets file. **Figure 12** summarizes the accuracy metrics.

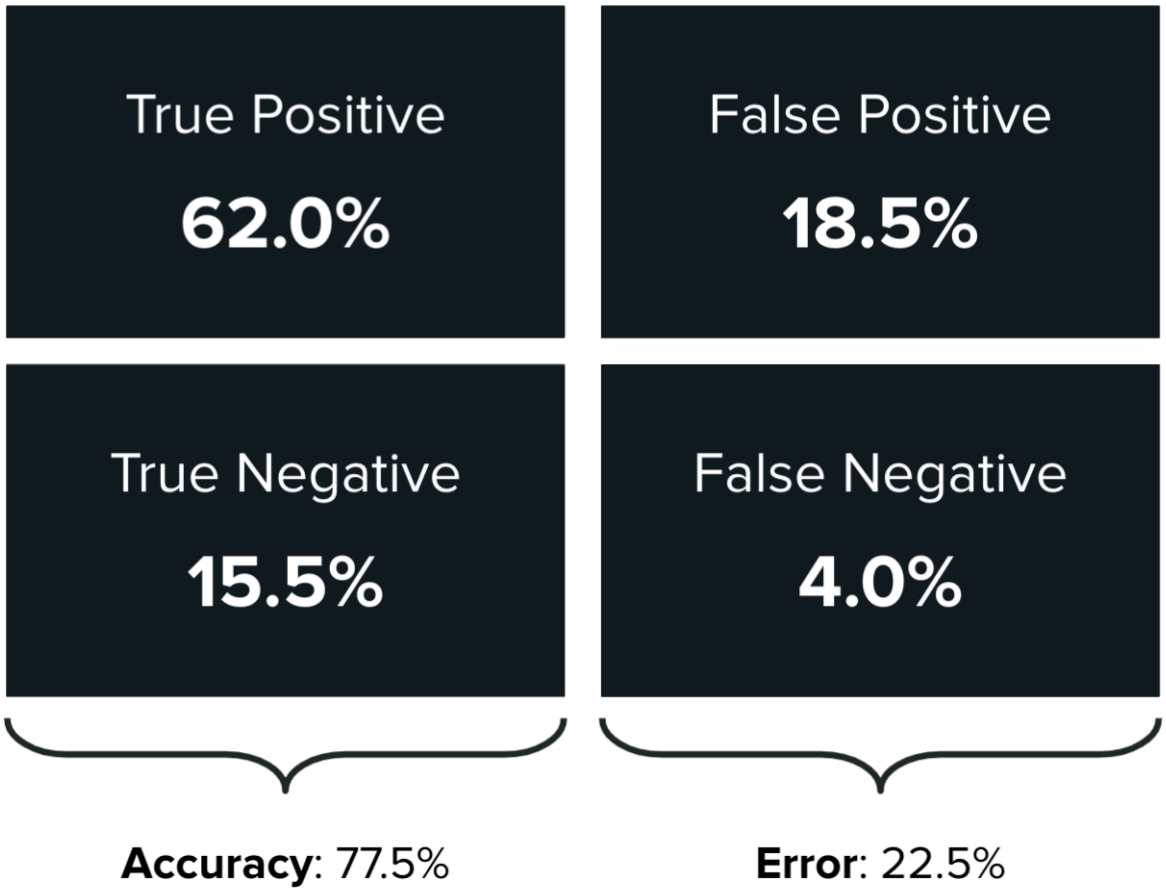


Figure 12: Accuracy metrics

Limitations

While the development of this proof of concept clearly proves the possibility to successfully monitor the occurrence of abnormal events through Twitter/Reuters data, our developed solution could be further improved to provide higher quality information.

When investigating the linguistics of scraped Tweets/headlines, we found that while the vast majority of Tweets/headlines are written in English, some Tweets/headlines were written in other alphabets. Consequently, further iterations should enable the processing of Tweets/headlines written in other languages such as Chinese. While this is not a significant limitation given that we can detect the vast majority of worldwide events, failure to have the

first movers advantage in some circumstances could hinder the ability to capitalise on a potential opportunity effectively.

Moreover, the algorithm is currently incapable of differentiating between past and present events. For example, the Tweet “There was a fire 30 years ago in Minatitlan” would not be understood as a past event. While this case will very rarely happen, we believe that this could be a next step to improve the model.

Finally, the model could not always distinguish events impacting other types of refineries, such as gold or sugar refineries. Although this barely affected the algorithm’s results, filtering out these refineries could partly reduce noise in the dataset.

Next Steps

While rendering this proof-of-concept marks a significant milestone, the journey to develop a commercially viable refinery event monitoring system is still incomplete.

Moving forward, we would encourage OilX to learn from our successes and shortfalls to develop a new solution based on a similar architecture. However, future iterations should focus on overcoming the proposed limitations.

Further improvements involve differentiating between present and past tense, analyzing sentiment to get an emergency score, and assessing the trustworthiness of the sources to assign a different weight to each tweet/headline in the clustering. Ideally, the algorithm should also be capable of predicting the impact on refinery capacity to maximize value for traders and analysts.