Project 5

Dan Chen, Ramiro Ramirez, Hitha Yeccaluri

Problem statement:
How can we make finding information via Twitter more convenient for victims of natural fires?

Background

In disasters, social media can be used for two major things: communication between individuals and as an “emergency management tool”

- Victims can share information about the situation, alert others, mark themselves as safe
- Volunteers can coordinate across distances easily and adjust their efforts based on real-time progress reports
- Officials in charge of emergency services can identify the type of help that is needed and where the aid needs to be concentrated

Projects like **Ushahidi** and **Twitcident** are dedicated to crowdsourcing information surrounding tragedies in real time and advancing crisis responses and protection/prevention efforts

Social Media is Already Making a Difference:

Hurricane Sandy's response teams were able to use the "power of collective knowledge" to gain a big-picture understanding of the impact and debunk misinformation

Hala Systems, a tech startup, uses their technology to detect airstrikes in the **Syrian civil war** and then goes through Facebook and WhatsApp to alert people in areas that will potentially be affected

In **Afghanistan** right now, it is a double-edged sword

International teams are dedicated to labeling and analyzing digital data generated during tragedies like **Hurricane Harvey** and the **2015 Paris terrorist attacks**.

Workflow

Data Collection

Decided parameters, found the best scraper

Cleaning

Data was collected by hand, we needed to comb through it

Labeling

Figured out how to label each data point in such a large set

Modeling

Tried two vectorizing methods in each classification model

Evaluation

Evaluated models and which would be best for what situation

Collecting, Extracting, Cleaning

How we scraped

- Set location limits to 160km around Napa, CA with snscrape
- Focused on the second half of 2020
- Used multiple keywords based on the paper [Endsley et al. 2013]

Raw data: 108,825 tweets

Features we extracted

- Keyword search associated with tweet
- Created column with username
- Is the user verified?
- Does this tweet appear more than once?
- “label” column - is this tweet relevant?

What we cleaned

- Kept ‘user’, ‘date’, ‘content’, ‘user_location’ and ‘id’ columns
- Grouped locations together (ex. ‘CA’, ‘California’ and ‘California, USA’ all mean the same thing, but people use all three)

Final shape: (108825, 10)

Labeling Process

With over 108,000 tweets, hand labeling each individual tweet was not plausible.

Inspired by a 2017 study about mining informative words from tweets during disasters, we wrote a function that would classify each tweet as relevant/irrelevant based on whether or not it contained a certain word

Our process was two-fold:

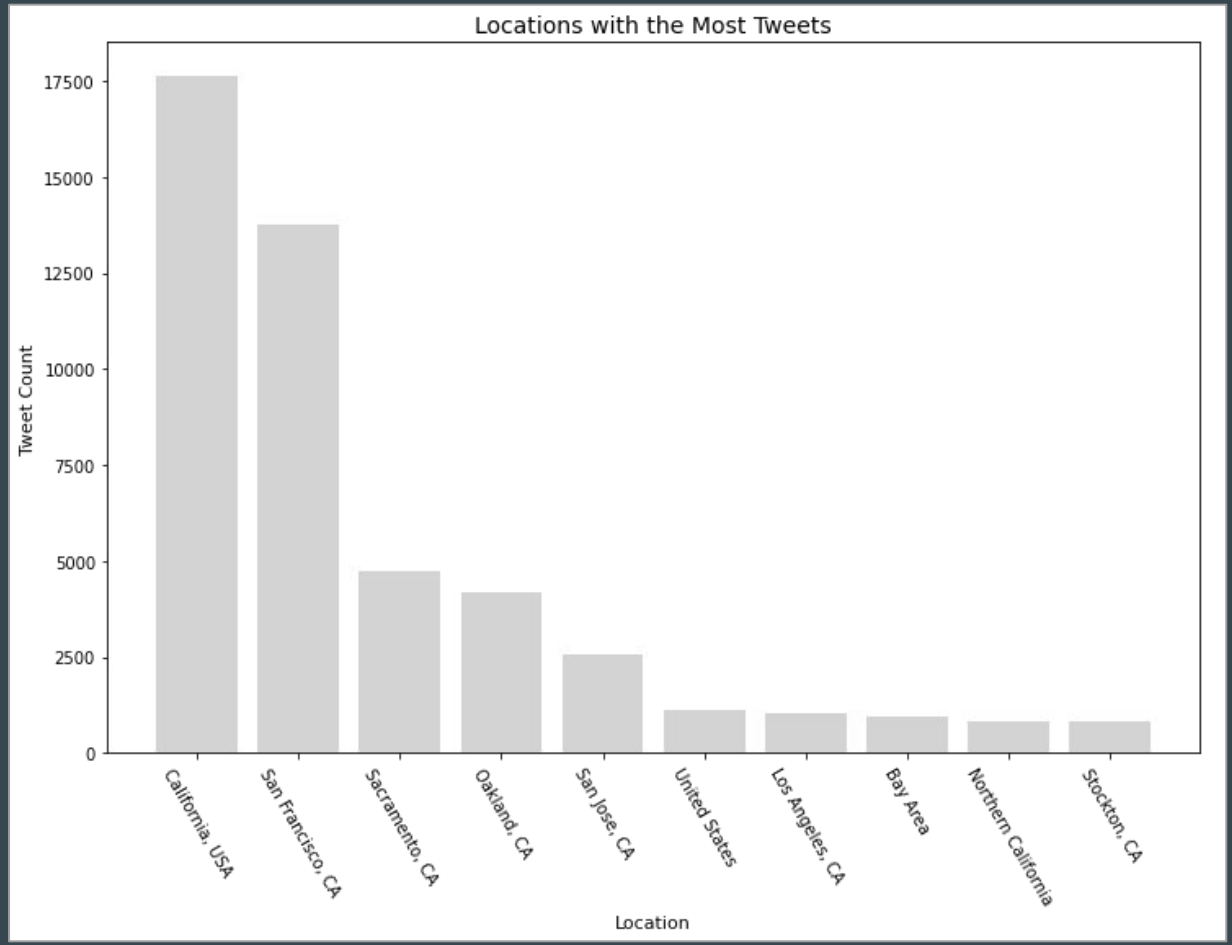
1. Searched through the first few thousand data points and took note of words that appeared frequently across tweets we deemed “relevant” to the specific situation (CA wildfires in 2020)
2. Searched the article for general words the authors suggested for disasters

Wrote the function to check each tweet against the list and label it “1” for relevant and “0” for irrelevant. While this function did well in labeling, we acknowledge that we did lose some of the nuance that may have been included in hand-labeling.

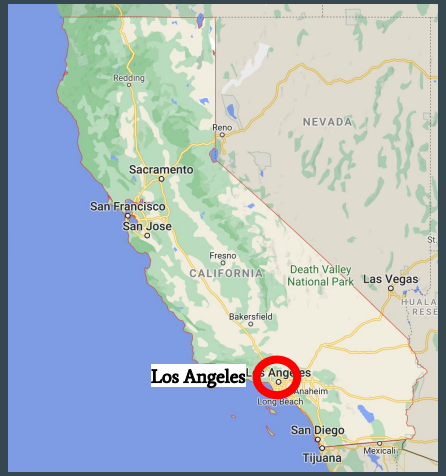
Locations

Difficult to track because many Twitter users do not enter their actual location in the ‘Location’ section of their profile and do not want their geolocation to be used.

Besides the “United States”, all the top 10 locations are in California.

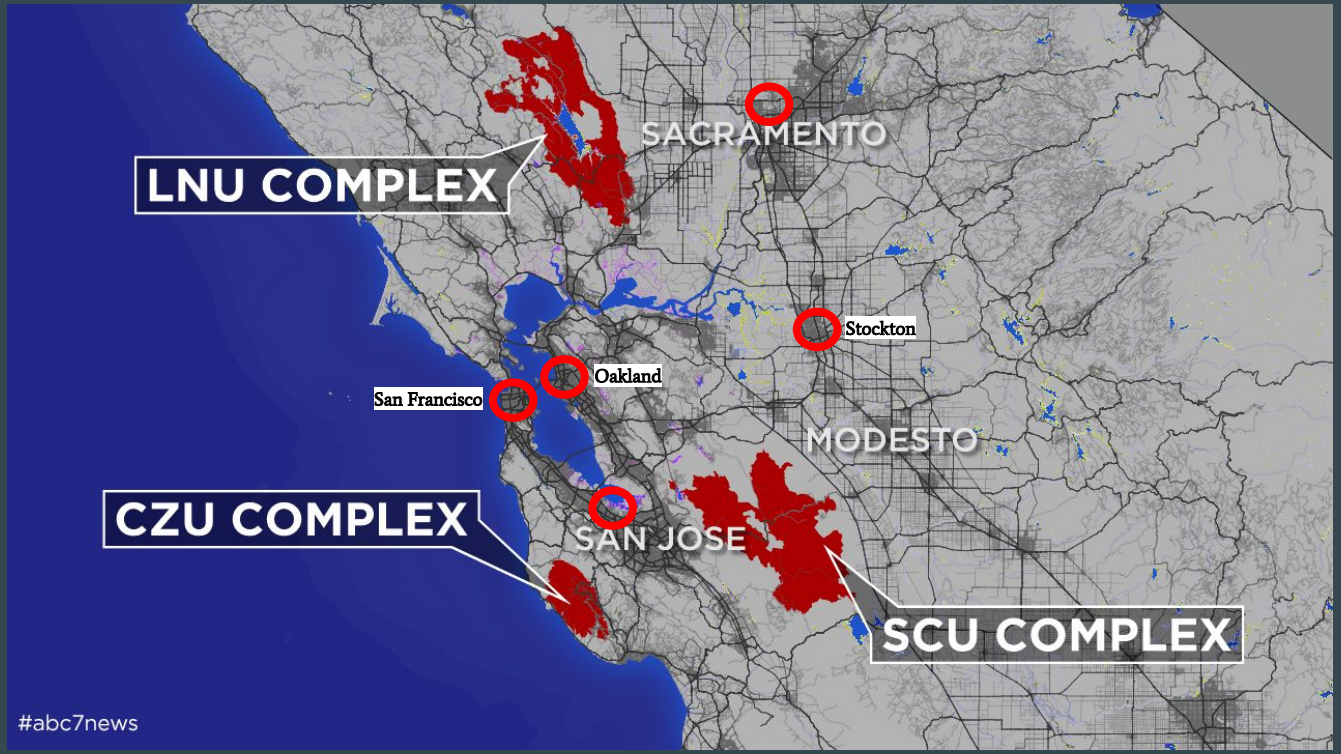


Spread of Locations



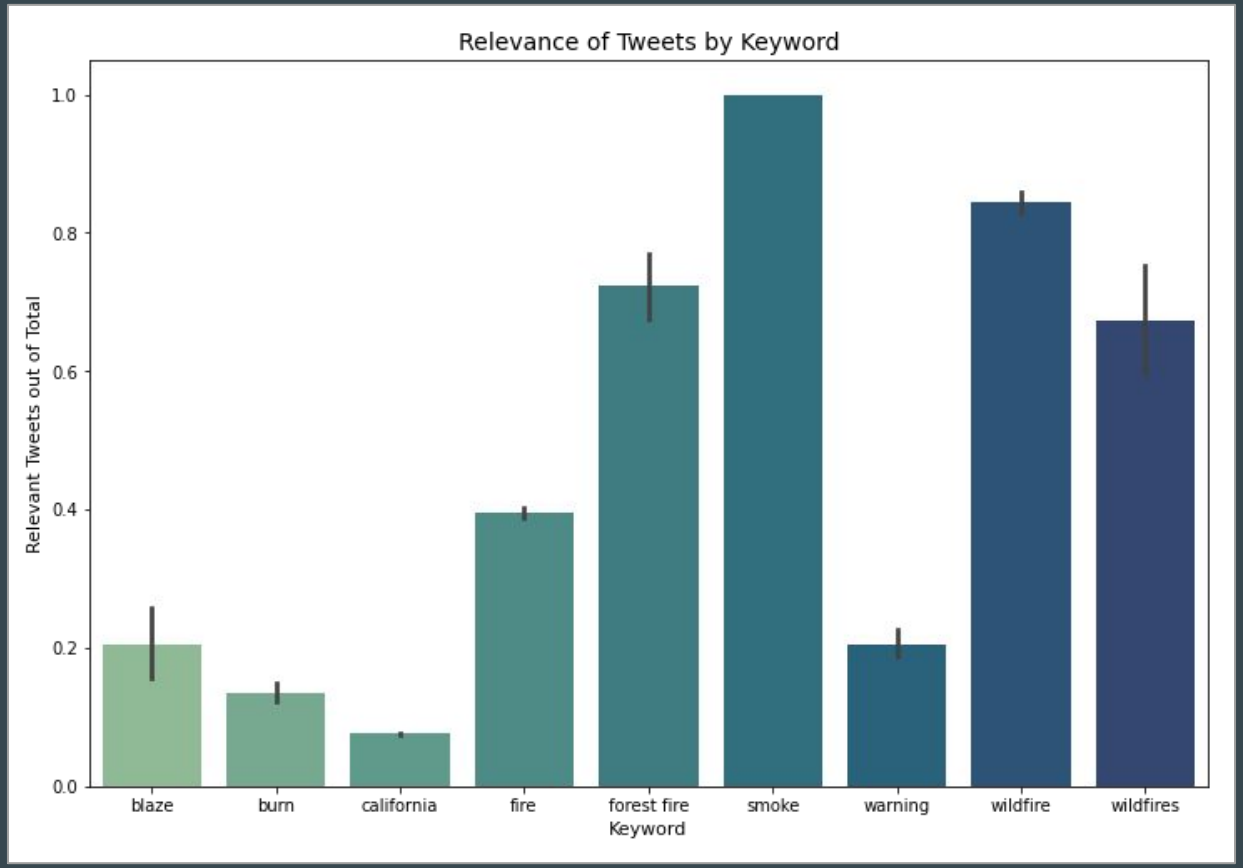
Red circles indicate cities that were in the previous chart.

Labeled complexes refer to the three major fires.



Keywords

We used nine different keywords when scraping for tweets - they were all words commonly associated with the CA fires.



Models Tested

Baseline to beat: 0.7685

Logistic Regression

KNeighborsClassifier

Decision Tree

Random Forest

SVC

LinearSVC

Multinomial Naive-Bayes

CountVectorizer vs. TF-IDF Vectorizer

We ran each of our models with the data vectorized in one of two ways. While there seems to be slight favoritism towards TF-IDF among everyone in machine learning forums, both methods have their benefits.

CountVectorizer functions by marking words as more statistically significant the more that they appear, and can be apt at identifying the type of text it is vectorizing.

TF-IDF Vectorizer weights the word frequencies and prioritizes words that don't appear as frequently.

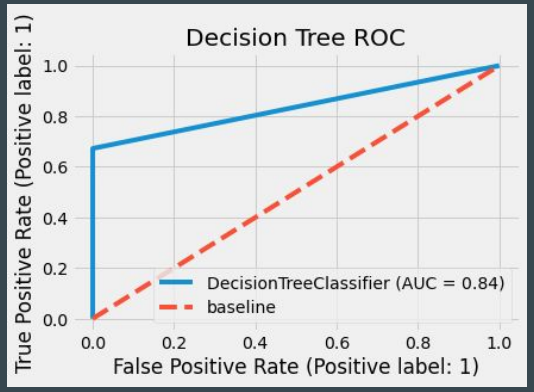
We thought it would be worth exploring how each type performed in a binary classification model.

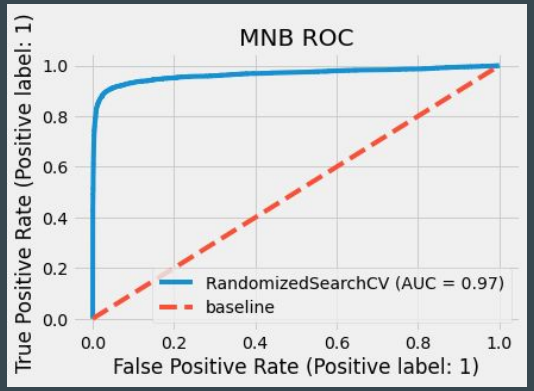
	CountVectorizer Train Score	CountVectorizer Test Score	TF-IDF Train Score	TF-IDF Test Score
Logistic Regression	0.9999	0.9814	0.9611	0.9998
KNeighborsClassifier	0.9964	0.9233	0.9163	0.8711
Random Forest	0.9648	0.9636	0.9999	0.9508
Decision Tree	0.9222	0.9240	1.0	0.9890
Support Vector Classifier	0.98897	0.9632	0.9934	0.9740
LinearSVC	0.9997	0.9884	0.9991	0.9864
Multinomial Naive-Bayes	0.9364	0.9321	0.9441	0.9426

Highlighted models had the lowest amounts of over/underfitting

ROC Curves and AUC Scores

- MNB has an almost perfect AUC score of 0.97. This means that when a tweet is randomly selected, the MNB model will be able to distinguish whether it is relevant or irrelevant 97% of the time.
- MNB performed better than DT in this metric; MNB identified true positives better than DT.





Metrics

We chose to look at the accuracy, precision and recall.

Accuracy: of all the classifications, how many tweets were correctly classified as 'relevant' and 'not relevant'?

Precision: of the tweets that the model labeled 'relevant', how many are actually relevant?

Recall: of the tweets that are actually relevant, how many are caught by the model?

	Accuracy	Precision	Recall
LR	0.9814 / 0.9610	0.9804 / 0.9440	0.9386 / 0.9129
KNN	0.9233 / 0.8711	0.9801 / 0.7307	0.6831 / 0.7021
RF	0.9648 / 0.9636	0.9967 / 0.9694	0.8456 / 0.9038
DT	0.9240 / 0.9891	0.9996 / 0.9888	0.6725 / 0.9644
SVC	0.9632 / 0.9740	0.9987 / 0.9995	0.8422 / 0.8884
LSVC	0.9884 / 0.9864	0.9963 / 0.9989	0.9535 / 0.9425
MNB	0.9321 / 0.9427	0.8127 / 0.8856	0.9191 / 0.86396

Scores are listed by CountVectorizer pre-processing / TF-IDF pre-processing.

Model Analysis

Findings:

Running a variety of models gave us a wide range of choices. The models with the highest metrics are:

- Accuracy:
  - TF-IDF DT: 0.9891
- Precision:
  - CountVectorizer DT: 0.9996
- Recall:
  - TF-IDF DT: 0.9644

- Why prioritize **accuracy**?
  - Accuracy is important for understanding how well the model is doing as a whole and is a fairly intuitive metric, making it ideal to use when speaking with non-technical audiences
- Why prioritize **precision**?
  - When a situation is not dire and people just need to be pointed in the direction of resources and info, good precision would be best. *This is what we want for this model in this context.*
- Why prioritize **recall**?
  - Optimizing this would ensure that people see only the good tweets, absolutely nothing else. For severe situations, this would be the metric of concern.

Recommendations, Conclusions and Expansions

Depending on the situation, we can choose the model that will optimize for precision, recall or accuracy. Based on the context of the 2020 CA fires, we would recommend using the model that has the best precision. The metric to optimize for would depend on the severity of the fire.

Ideally, this model would be implemented on a livestream of tweets, sorting them as relevant/irrelevant in real time in one convenient location. This would make information much more accessible for people who a) need answers and/or b) are not very tech-savvy and don't know how to use social media well.

However, things like this would be reliant on a stable Internet connection, which is not always guaranteed in disaster zones. Making sure it would still be useful in low-connectivity areas should be a top priority.