Using NLP to quickly curate data

Unsupervised Topic Modeling

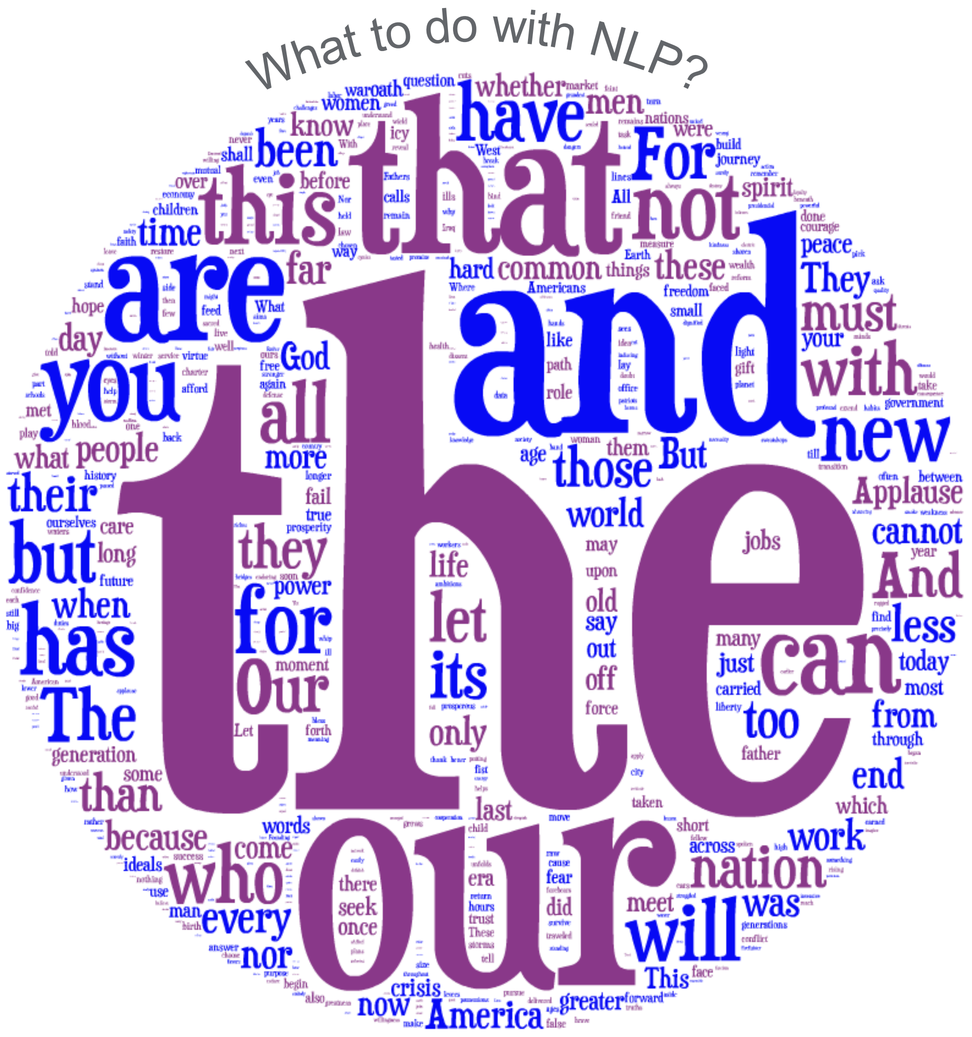
2/10/2021

Nathaniel Rigoni

LOCKHEED MARTIN

Overview

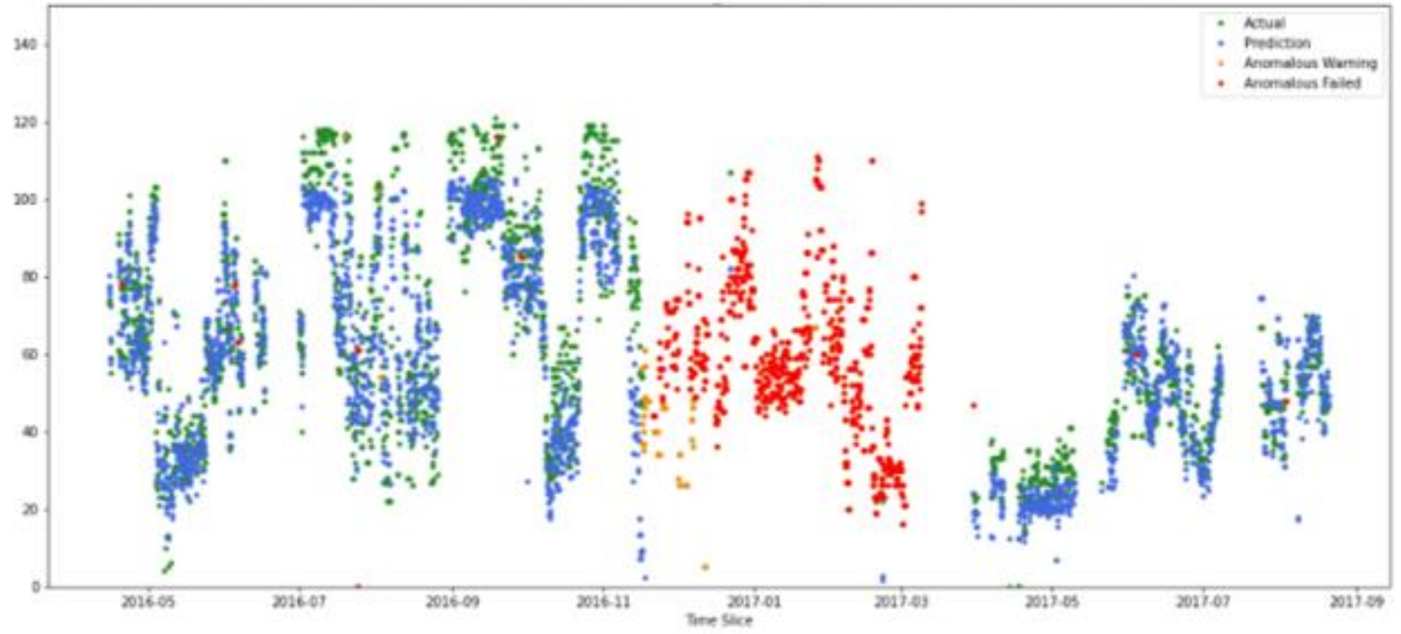
- What is a Virtual Sensor?
- Virtual Sensing workflow
- Curating data
- Word2Vec/Doc2Vec
- Term Frequency Inverse Document Frequency (TFIDF)

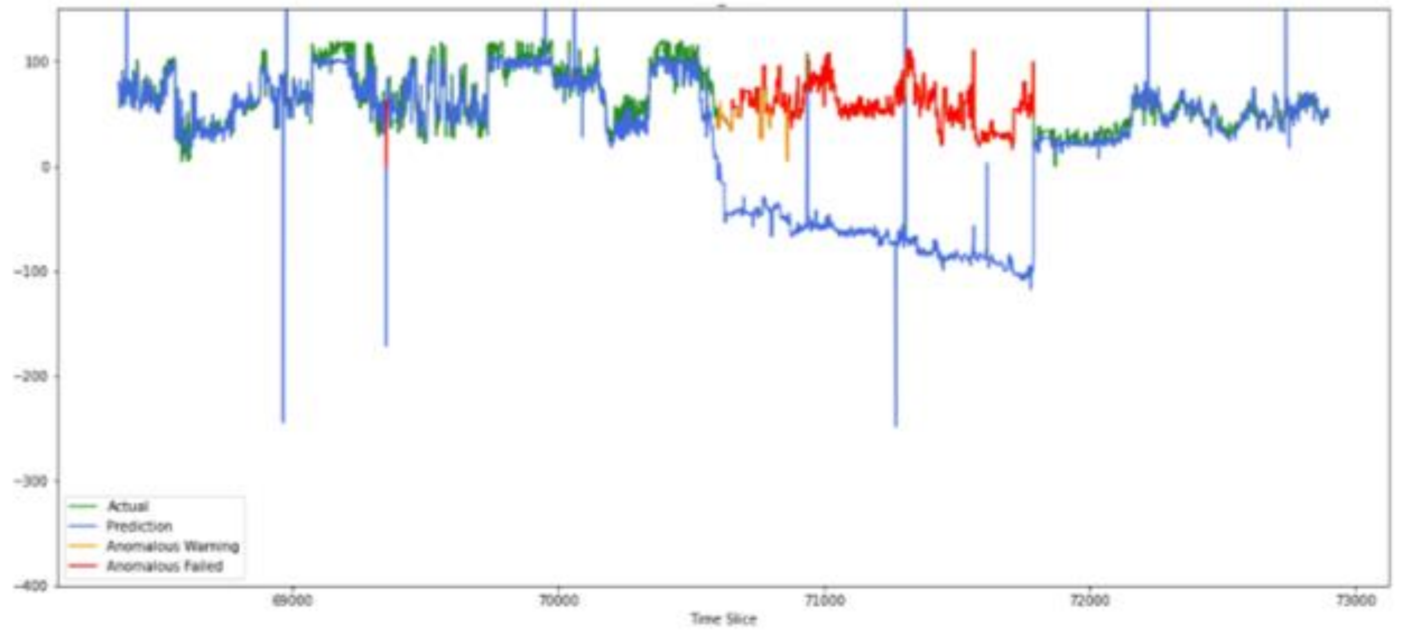


Use case

Develop method of identifying ground truth for Virtual Sensor models.

A Virtual Sensor model is trained to predict the nominal behavior of a system. So when a system is not behaving nominally the virtual sensor will identify this activity.



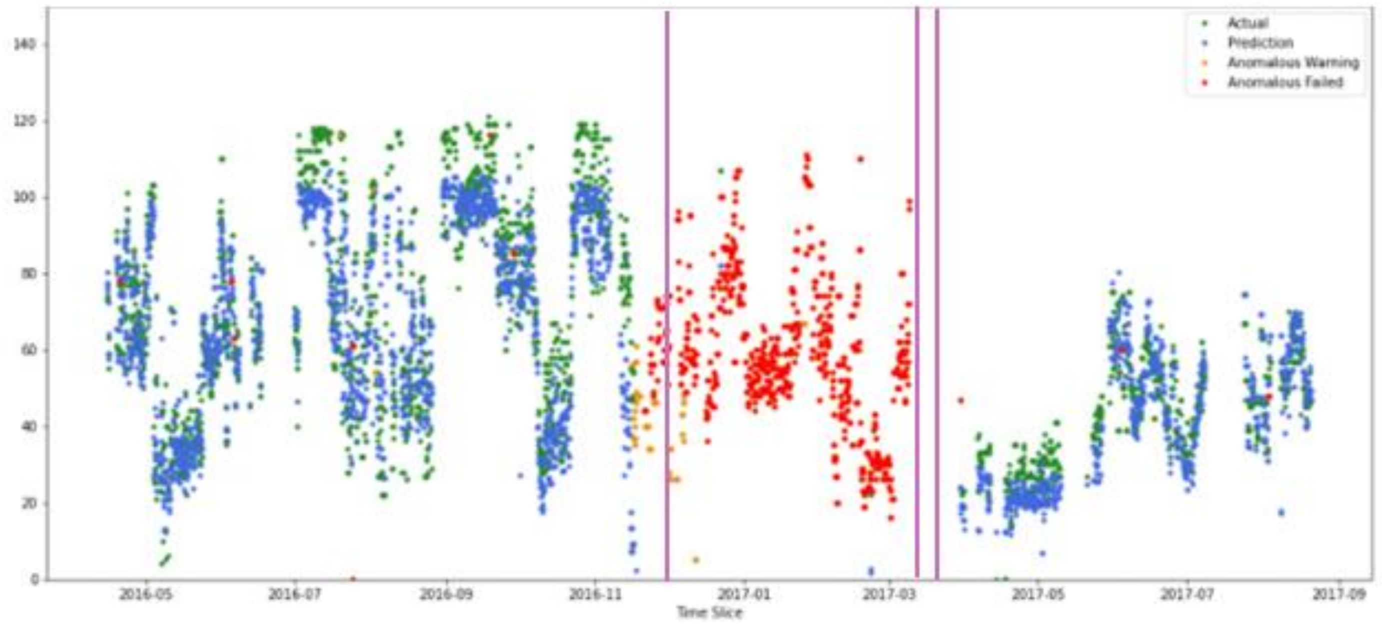


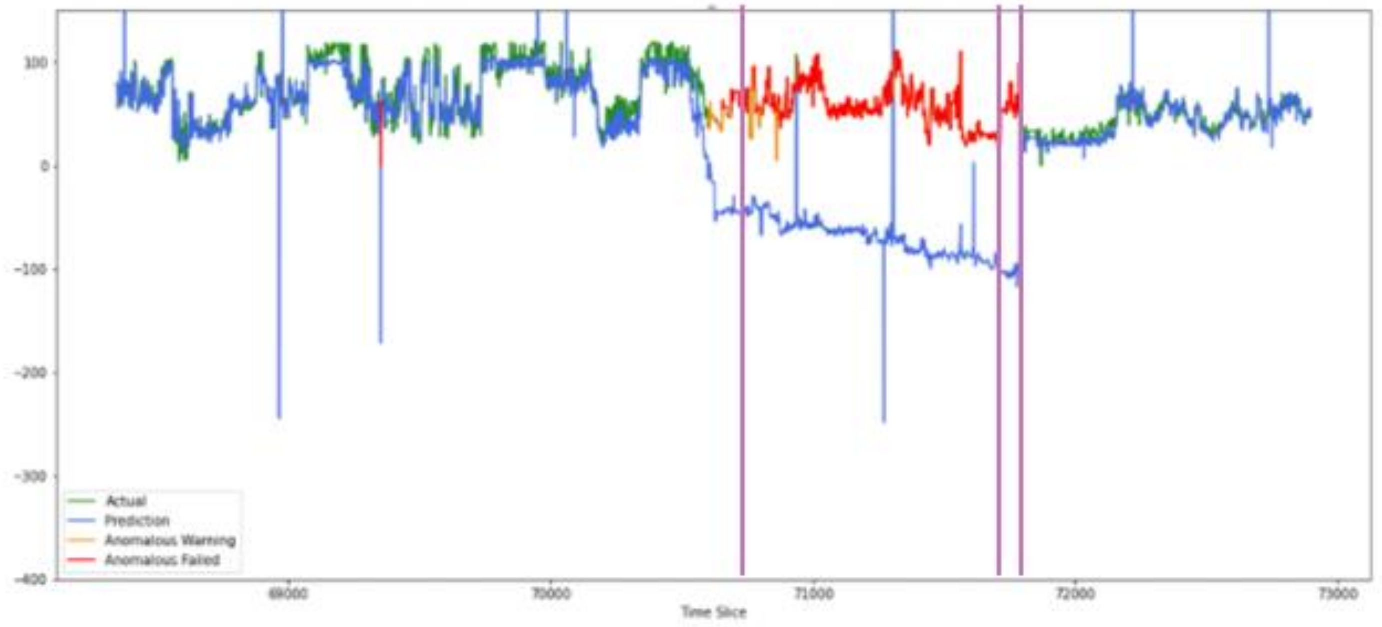
Work Flow to build a VS model

We want to remove:

- Noise
- **The data that is associated with maintenance events**

This is where the NLP comes in





Finding instances of failure/maintenance

Methods:

- Word2vec
- Doc2Vec
- TFIDF
- Bigram/Trigram

We don’t have to find every instance of maintenance. Just enough to make sure that more than 90% of the data is nominal operation.

Bigrams/Trigrams

Phrase analysis is a method of examining words that often occur next to each other in documents.

- Examines corpus for word co-occurrence
- Combines words that commonly occur together into a single word separated by and underscore

Creating these and embedding them into a corpus before training a word2vec model allows for the model to overcome word/context granulation.

1+1=3

Main Rotor Blade

=

Main_Rotor_Blade

Word2Vec/Doc2vec

- Word2Vec is a vectorized encoding at the word level for all words present in the corpus.
- It is built by creating a model that tries to predict context words that surround a word in a skip gram or continuous bag of words.
- Its output is the hidden layer embedding from the neural net for each word in the corpus
- Doc2vec adds an identifier for the document such as the index of the record. An embedding is also learned for this index.

Continuous Bag of Words problem

The red dog ran down the lane

The red dog ___ down the lane

Skip Gram problem

The red dog ran down the lane

___ ___ ___ ran ____ ___ ____

How is this useful?

- Word embeddings will be optimized to represent the words that commonly surround them.
- If two words are synonymous they will usually be surrounded by the same words
- When trained with bi-grams and tri-grams this method can identify compound word similarities and acronyms
- With doc2vec we can now compare the similarity of the learned embedding for each index of documents

Continuous Bag of Words problem
The red dog ran down the lane
The red dog ___ down the lane

Skip Gram problem
The red dog ran down the lane
___ ___ ___ ran ____ ___ ____

How is this useful?

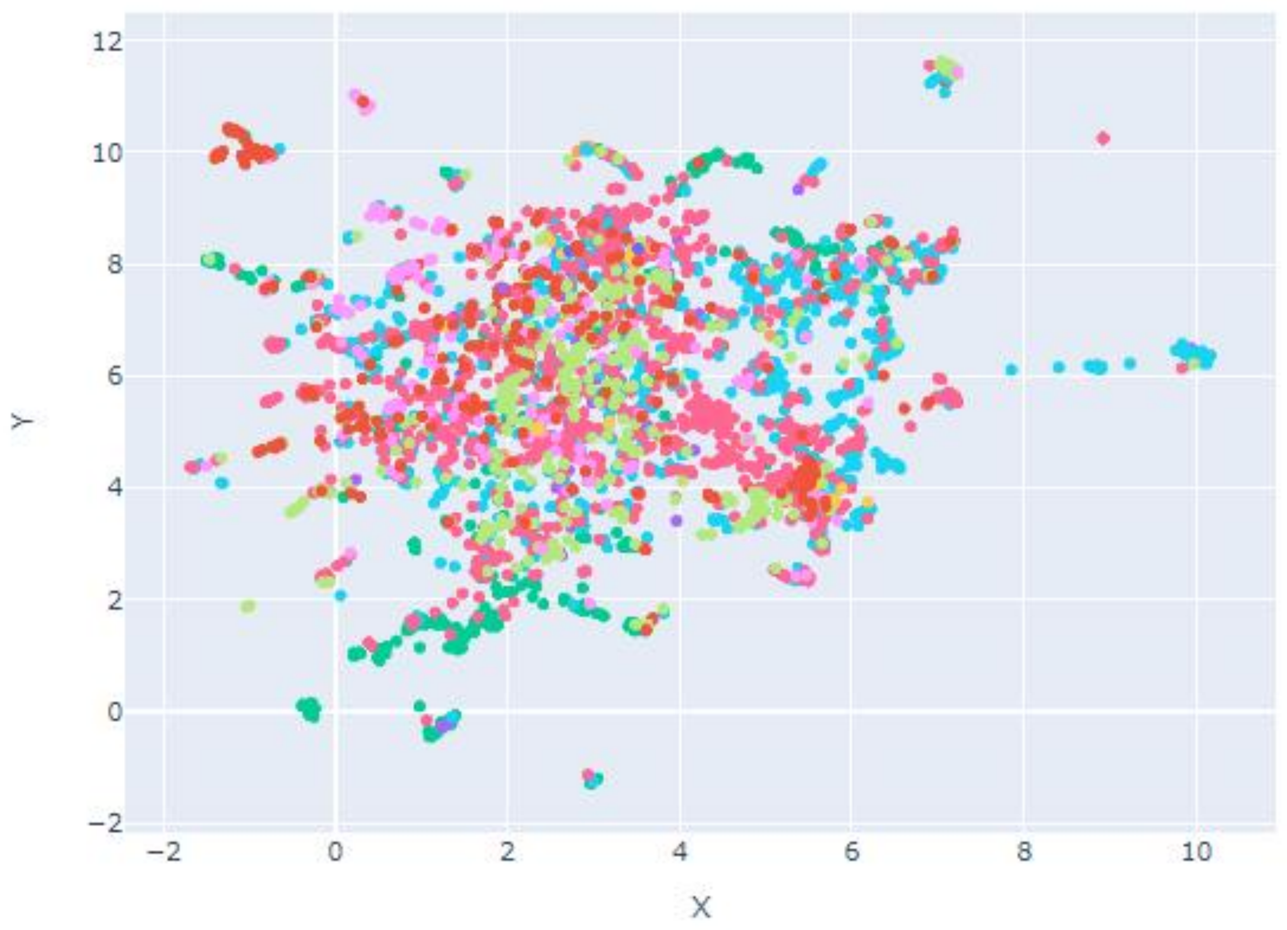
- Word vectors can be compared for similarity of words. This means we can find the synonyms of words or the most related words to a given word.
- Document vectors can be clustered in N-Dimensional space to give us Topics of documents in the whole collection of documents.

Related words:
Main_Rotor_Blade
MRB
Red
Blue
Black
Yellow

Related documents:
Black MRB Cracked
Yellow MR Blade delaminated
Blade delaminated

Doc2Vec clusters

- Colors represent clusters in N-dimensional space.
- Clusters are created using HDBSCAN
- Clusters represent topics in a language space and are broken down in a hierarchical method
- Clusters are points that are dimensionally close to each other
- Items in a cluster are both linguistically similar and topically similar



Utilizing Word2Vec/Doc2vec in search

This comparison is usually done using the Cosine Similarity:

$$cosine\ distance = \frac{\cos^{-1}\left(\frac{A * B}{||A||\ ||B||}\right)}{\pi}$$

$$cosine\ similarity = 1 - cosine\ distance$$

A – the aggregated vector of all of the search terms

B – the aggregated vector of any one record

w2v.most_similar("fuel")

```
[('oil', 0.8083397746086121),
('jp', 0.7940911054611206),
('filterseperator', 0.7511616349220276),
('lube', 0.7367134690284729),
('fo', 0.7269341945648193),
('fule', 0.6887214779853821),
('coalescer', 0.680597186088562),
('flube', 0.679574191570282),
('tank', 0.677828848361969),
('filltransfer', 0.6747710704803467)]
```

Utilizing Word2Vec/Doc2vec in search

Drawbacks:

- We can see similar systems end up as synonyms in our model and inhibit our ability to search for the correct system.

Fix:

- Use negative terms in search to remove unwanted words.

(GTG and not GTM)

w2v.most_similar("fuel")

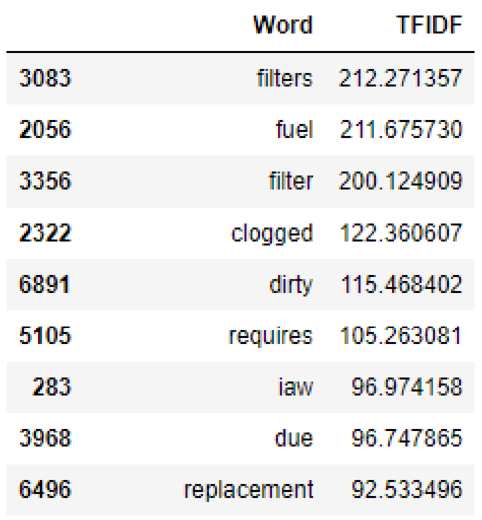
```
[('oil', 0.8083397746086121),
('jp', 0.7940911054611206),
('filterseperator', 0.7511616349220276),
('lube', 0.7367134690284729),
('fo', 0.7269341945648193),
('fule', 0.6887214779853821),
('coalescer', 0.680597186088562),
('flube', 0.679574191570282),
('tank', 0.677828848361969),
('filltransfer', 0.6747710704803467)]
```

TFIDF

- Examines the importance of words by measuring there occurrence in the individual document vs occurrence in all documents.

- TFIDF on search results help summarize the contents

Search phrase: “fuel filter"



	Word	TFIDF
3083	filters	212.271357
2056	fuel	211.675730
3356	filter	200.124909
2322	clogged	122.360607
6891	dirty	115.468402
5105	requires	105.263081
283	iaw	96.974158
3968	due	96.747865
6496	replacement	92.533496

TFIDF based on the results that have a cos-sim>0.6

Curating VS data example

The steps to rapid curation are as follows:

1. Build word2vec/doc2vec model (with or without bigrams/trigrams)
2. Embed vectors of model into data
3. Perform semantic search by getting the aggregated vector of all search terms and then comparing the cosign similarity of each record vector to your search vector.
4. Analyze topics of results, and remove results with unrelated topics
5. Use results as ground truth for model

Demo
On
Amazon Reviews

LOCKHEED MARTIN