Problem Description

I lead a team of software testers who have been writing test cases for several years and now have over 1,000 cases in our history. Our test tests the software on Large Industrial Drilling Machines and the automation of those machines. Test Cases are a record of our testing process and represent a significant investment of work. One of our consistent struggles is trying to identify old test cases that might cover the behavior we are trying to test with the newest software. If we could successfully identify and reuse old cases, it would greatly speed up our test writing process and improve the quality of our testing program.

Currently our testing framework has a text matching search, which will match text searches to the name of an old test case. However, the name of our test cases have been copied from the name of the software change they are testing and are therefore not even close to a good summary of the case in question.

There is a need for a tool that will process the name, objective and actual test steps from our library of test cases and compare it to a description of the problem being tested to look for previous cases which might come from the same category or test similar parts of the software.

The goal of this project is to gather and process the test cases into a text format which can be encoded using a word count vectorizer such as TfidfVectorizer from Sklearn. This type of vectorizer was chosen because of mixed results in testing with a LLM vectorizer like Bert. I think that the limitation are related to the fact that our cases are almost entirely jargon and LLM which aren't specifically trained on software testing in the mining industry struggle to make sense of the text. It is difficult to go beyond a guess because the other limitation of vectorizers like Bert are that the results are incomprehensible and almost impossible to troubleshoot.

Once the cases are lemmatized (enrich the meaning of each word by reducing variants to single entries) and vectorized, an SVD model is used to reduce the 10,000+ features from the set into a lower dimension matrix which captures at least 95% of the variance in the original model.

The lower dimensional matrix is then used as the input to a k-means clustering algorithm which attempts to group them by category. When a search term is provided, the same process is followed to vectorize the term and then label for the search term is predicted. From the predicted category, the cosine similarity is used to compare with the items in that category and then the top 5 cases are returned to the user.

There is parameter tuning applied to the SVD model iteration and the K-means clustering. I used both Silhoutte Score and Calinski-Harabasz score to evaluate the quality of the

clustering with each number of clusters though Calinski-Harabasz score seems to provide a more relaiable tuning metric.

EDA procedure

Clean and preprocess

Imports

In [1]:
```
import yaml
import pandas as pd
import numpy as np
import re
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.decomposition import TruncatedSVD
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import Normalizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import calinski_harabasz_score
from sklearn.metrics import silhouette_score
```

In [2]:
```
# Download latest resources
#nltk.download('punkt')
#nltk.download('wordnet')
```

In [3]:
```
with open(r'assets\cases.yaml') as f:
    raw_data = pd.DataFrame(yaml.safe_load(f.read()))
```

In [4]:
```
raw_data
```

Out[4]:

	owner	updatedBy	updatedOn	majorVersion	priority
0	myles.ross@us.atlascopco.com	usstwilk	2022-09-02T18:20:59.620Z	1	Normal
1	myles.ross@us.atlascopco.com	usstwilk	2022-09-02T18:21:00.695Z	1	Normal
2	myles.ross@us.atlascopco.com	usstwilk	2022-04-24T20:21:23.264Z	1	Normal
3	myles.ross@us.atlascopco.com	usstwilk	2022-04-24T20:21:23.273Z	1	Normal
4	myles.ross@us.atlascopco.com	usstwilk	2022-04-24T20:21:23.224Z	1	Normal
...	...	...	...	...	...
1732	JIRAUSER51128	JIRAUSER51128	2023-09-12T12:40:12.065Z	1	Normal
1733	usstaikm	usstaikm	2023-09-14T16:41:02.932Z	1	Normal
1734	usstwilk	usstwilk	2023-09-12T21:10:48.684Z	1	Normal
1735	JIRAUSER51128	JIRAUSER51128	2023-09-14T17:50:10.234Z	1	Normal
1736	JIRAUSER51128	JIRAUSER51128	2023-09-14T17:50:06.009Z	1	Normal

1737 rows × 23 columns

Data Description

Data is the entire text from 1732 test cases that have been written for our software over the last few years. The data dump has been downloaded from the HTML based website and converted to a Yaml format saved as a file. There are variety of helpful but mostly non helpful features. For example, Created Date, UpdatedBy, owner, major version and many others are not relevant to helping find relevant test cases. Further, the bulk of the actual text data is located in the testCript column and is stored as a further nested data structure with even more extranous data.

The first step is to only look at test cases with the status "Ready to Run" or "Completed" since all other statuses ndicate cases that are not complete or are deficient in some way which we would not want to review and learn from. At the same time, only the columns which are desired are kept. Namely testScript, key, objective and status.

Key is the best identifier for each test case and is unique per test case. Objective contains a summary fo the software change and will help in grabbing some of the software descrioton that might not make it into the test script. testScript is the text added by the testing team which will contain the bulk of the information added by our team and therefore most of the valuable data in the analysis.

In [5]:
```
filtered_data = raw_data[raw_data['status'].isin(['Ready to Run', 'Completed'])][['
filtered_data
```

Out[5]:

	key	testScript	objective
0	ADSRTE-T3	{'id': 97, 'type': 'STEP_BY_STEP', 'steps': [{...	NaN
1	ADSRTE-T5	{'id': 208, 'type': 'STEP_BY_STEP', 'steps': [...	Test AD2 turn air on when feeding down in retr...
2	ADSRTE-T6	{'id': 209, 'type': 'STEP_BY_STEP', 'steps': [...	Test ADSRTE-1245 No engine throttle control (T...
3	ADSRTE-T7	{'id': 210, 'type': 'STEP_BY_STEP', 'steps': [...	Test ADSRTE-1246 Rigs crash without navigation...
4	ADSRTE-T9	{'id': 212, 'type': 'STEP_BY_STEP', 'steps': [...	Test ADSRTE-1220 Drill String Guarding not in ...
...	...	...	...
1727	ADSRTE-T1941	{'id': 15075, 'type': 'STEP_BY_STEP', 'steps':...	Assert logs generated at after reboot.
1728	ADSRTE-T1942	{'id': 15076, 'type': 'STEP_BY_STEP', 'steps':...	Assert logs generated at after reboot.
1730	ADSRTE-T1944	{'id': 15078, 'type': 'STEP_BY_STEP', 'steps':...	It would be nice if we could know what version...
1731	ADSRTE-T1945	{'id': 15079, 'type': 'STEP_BY_STEP', 'steps':...	In 5.6 it was found that the event for a TUM e...
1732	ADSRTE-T1946	{'id': 15088, 'type': 'STEP_BY_STEP', 'steps':...	<img src="/secure/attachment/1365017/1365017_i...

1344 rows × 3 columns

In [6]:
```
# Keep only the "steps" from the testScript column, removing unneeded metadata abou
filtered_data['steps'] = filtered_data['testScript'].apply(lambda d: d.get('steps')
```

In [7]:
```
# Expand the DataFrame on steps
expanded_data = filtered_data.explode('steps', ignore_index=True)
# Remove test cases with no test steps
expanded_data = expanded_data.dropna(subset=['steps'])
    # Note, some of the cases were created without any steps at all and should tech
```

In [8]:
```
# Pull out testData, expectedResult and description from the test steps
expanded_data['testData'] = expanded_data['steps'].apply(lambda d: d.get('testData'
expanded_data['expectedResult'] = expanded_data['steps'].apply(lambda d: d.get('exp
expanded_data['description'] = expanded_data['steps'].apply(lambda d: d.get('descri
# Drop steps which are empty
expanded_data = expanded_data.dropna(subset=['description'])
    # Note, The cases with no steps were dropped in the previous cell, but even wit
        # some of the steps were left empty and they are removed here specifically,
```

Cleanup DataFrame

In [9]:
```
# Rename the unclear column names
expanded_data.rename(columns = {'testData':'Comment',
                                'expectedResult' : 'Expected Result',
                                'description' : 'Step',
                                'objective' : 'Objective',
                                'key' : 'Key'}, inplace = True)
```

In [10]:
```
# Keep only the relevant data from the intermediate steps getting here.
relevant_data = expanded_data[['Key','Objective','Step','Expected Result','Comment'
relevant_data
```

Out[10]:

	Key	Objective	Step	
0	ADSRTE-T3	NaN	Navigate to M5.3.6 on RCS and Open current Dat...	DBI should
1	ADSRTE-T3	NaN	In RCS, change the value of "Limit, Drill not ...	
2	ADSRTE-T3	NaN	Re-open database file	
3	ADSRTE-T5	Test AD2 turn air on when feeding down in retr...	Start AD2	
4	ADSRTE-T5	Test AD2 turn air on when feeding down in retr...	After hole drilling complete, increase RT to b...	R
...	...	...	...	
7991	ADSRTE-T1944	It would be nice if we could know what version...	Follow the following path to the latest RCS 5 ...	git_sub
7992	ADSRTE-T1945	In 5.6 it was found that the event for a TUM e...	Insert USB drive into...	< drive<
7993	ADSRTE-T1945	In 5.6 it was found that the event for a TUM e...	Navigate to System - Administration - ...	The T
7994	ADSRTE-T1945	In 5.6 it was found that the event for a TUM e...	Remove the USB drive....	Th
7995	ADSRTE-T1946	<img src="/secure/attachment/1365017/1365017_i...	Get an active drill state, verify the rotation...	

7896 rows × 5 columns

In [11]:

```
def remove_tags(text):
    junk_text = ['None','<[^>]*>','&lt;','&gt;','nbsp']
    if isinstance(text, str):
        return re.sub('|'.join(junk_text), '', text)
    if isinstance(text, float):
```

```
        # nan is a float
        return ''
# Remove all the html tags and empty data from the dataframe
clean_data = relevant_data.applymap(remove_tags).fillna(' ')
# Check for any empty cells that were missed and print the result (Should be empty
clean_data[clean_data.isna().any(axis=1)]
```

Out[11]:

	Key	Objective	Step	Expected Result	Comment

In [12]:
```
# Collapse back into test cases with multiple steps
def combine_rows(rows):
    steps = []
    for index, row in rows.iterrows():
        step_str = f"{row['Step']} {row['Expected Result']} {row['Comment']}"
        steps.append(step_str)
    return pd.Series({'Steps': '\n'.join(steps)})

# Collapse the DataFrame by the "Key" and "Objective" columns
collapsed_data = clean_data.groupby(['Key', 'Objective']).apply(combine_rows).reset
```

In [13]:
```
collapsed_data
```

Out[13]:

	Key	Objective	Steps
0	ADSRTE-T10	Test ADSRTE-1047 [ABD-574] ADPE stops with no ...	With manual jack button active on ops station,...
1	ADSRTE-T100	Create and Run a test that is closely similar ...	Start ADPE ADPE will begin going through the s...
2	ADSRTE-T1002	At BHP with 4.20 rev.2 it is being reported (A...	Connect the rig to the ACMS and use it as the ...
3	ADSRTE-T1003	When running mixed fleet in Kevitsa, it was di...	Log in to ACMS Client and connect with the ACM...
4	ADSRTE-T1004	Description from Jayden Lambert from Aus CC:Du...	Set the ACMS time zone to anything other than ...
...	...	...	...
1333	ADSRTE-T992	Seen in 5.3 rev.5The OD Obstacle Violation log...	Navigate to Setting - Rig - Obstacle Detection...
1334	ADSRTE-T994	Problem Statement:On the RRA client the lab is...	Set the IP address back to the correct value f...
1335	ADSRTE-T995	After RRA Service is started, it will function...	Load a 2000 hole drill plan to the rig and hav...
1336	ADSRTE-T996	Originally written for ADSRTE-4454: DDS - Mach...	Load {Latest RCS 5 Version} to a machine. ...
1337	ADSRTE-T997	Seen in 5.4.0138488 was checked into the ACMS ...	Allocate and De-allocate Drill 1 from each Off...

1338 rows × 3 columns

In [14]:
```
# Save the data to a file to speed up future iterations of the data processing whic
collapsed_data.to_pickle("cleaned_data.pkl")
```

Analysis

Model building and training

In [15]:
```
# Restore saved  data
data = pd.read_pickle("cleaned_data.pkl")
```

In [16]:
```
# Feature extraction
# Compress the datframe to one text column
text_frame = pd.DataFrame(data['Key'])
text_frame['Text'] = data['Objective'] + ' ' + data['Steps']
text_frame
```

Out[16]:

	Key	Text
0	ADSRTE-T10	Test ADSRTE-1047 [ABD-574] ADPE stops with no ...
1	ADSRTE-T100	Create and Run a test that is closely similar ...
2	ADSRTE-T1002	At BHP with 4.20 rev.2 it is being reported (A...
3	ADSRTE-T1003	When running mixed fleet in Kevitsa, it was di...
4	ADSRTE-T1004	Description from Jayden Lambert from Aus CC:Du...
...	...	...
1333	ADSRTE-T992	Seen in 5.3 rev.5The OD Obstacle Violation log...
1334	ADSRTE-T994	Problem Statement:On the RRA client the lab is...
1335	ADSRTE-T995	After RRA Service is started, it will function...
1336	ADSRTE-T996	Originally written for ADSRTE-4454: DDS - Mach...
1337	ADSRTE-T997	Seen in 5.4.0138488 was checked into the ACMS ...

1338 rows × 2 columns

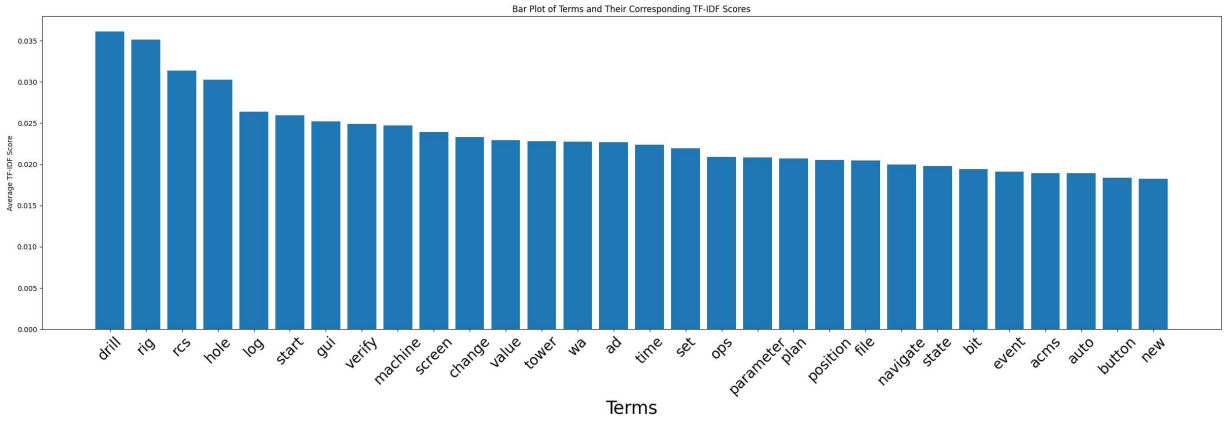
In [17]:
```
def lemmatize_and_letterize_text(text):
    digit_stripped = re.sub(r'\d', '', text)
    under_scored_rem = re.sub(r'_', ' ', digit_stripped)
    lemmatizer = WordNetLemmatizer()
    tokens = word_tokenize(under_scored_rem)
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
    lemmatized_text = ' '.join(lemmatized_tokens)
    return lemmatized_text

# Apply the function to the 'text' column
text_frame['lemmatized_text'] = text_frame['Text'].apply(lemmatize_and_letterize_te
```

In [18]:
```
# Vectorize the text into a weighted bag of words format
vectorizer = TfidfVectorizer(sublinear_tf=True,
                             stop_words='english')
X = vectorizer.fit_transform(text_frame['lemmatized_text'])
X
```

Out[18]:
```
<1338x7538 sparse matrix of type '<class 'numpy.float64'>'
        with 86382 stored elements in Compressed Sparse Row format>
```

In [34]:
```
# Visualizae the important terms
terms = vectorizer.get_feature_names_out()

# Calculate average TF-IDF scores for each feature
average_tfidf_scores = X.mean(axis=0).tolist()[0]

# Create a DataFrame
df = pd.DataFrame({'term': terms, 'average_tfidf': average_tfidf_scores})

# Sort by average TF-IDF scores in descending order
```

```
df_sorted = df.sort_values(by='average_tfidf', ascending=False)

# Select top N features with the highest average TF-IDF scores
top_n = 30
x = df_sorted['term'].head(top_n)
y = df_sorted['average_tfidf'].head(top_n)

# Create a bar plot using matplotlib
plt.figure(figsize=(30, 8))
plt.bar(x, y)
plt.xlabel('Terms', fontsize=25)
plt.xticks(fontsize=20)
plt.ylabel('Average TF-IDF Score')
plt.title('Bar Plot of Terms and Their Corresponding TF-IDF Scores')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```



Dimensionality Reduction

Current frame has ~7,500 features and will be hit hard by the curse of dimensionality.

In [20]:
```
# Dimensionality reduction parameter tuning
# Find the optimal number of components for TruncatedSVD
explained_variance_threshold = 0.95
n_components = 50 # Starting point
explained_variance = 0 # Init
best_n_components = 0 # Init

while explained_variance < explained_variance_threshold:
    print(f'Trying with {n_components} components')
    svd = TruncatedSVD(n_components=n_components)
    svd.fit(X)
    explained_variance = np.sum(svd.explained_variance_ratio_)
    print(f'Explained variance: {explained_variance:.2f}')

    if explained_variance < explained_variance_threshold:
        n_components += 50  # Increment the number of components and try again
    else:
        best_n_components = n_components
```

```
print('Optimal number of components:', best_n_components)
```

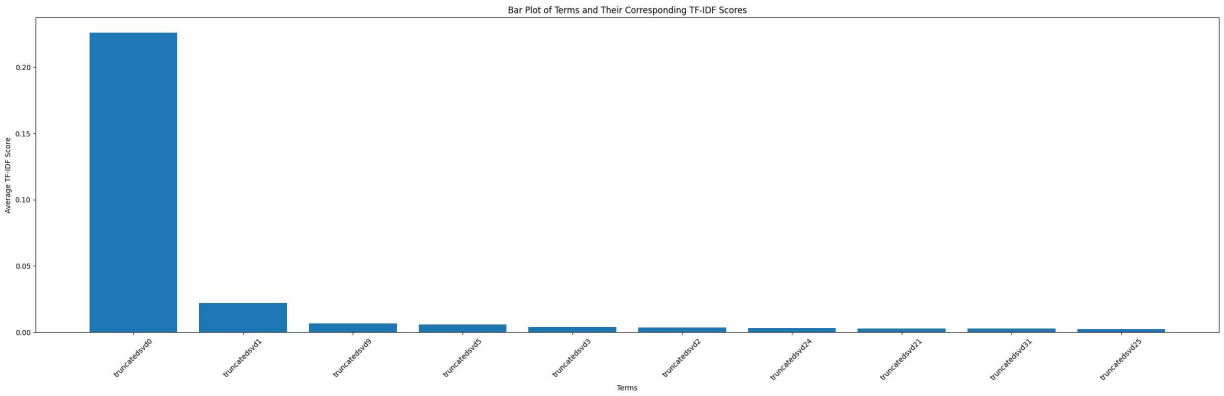
```
Trying with 50 components
Explained variance: 0.28
Trying with 100 components
Explained variance: 0.40
Trying with 150 components
Explained variance: 0.48
Trying with 200 components
Explained variance: 0.55
Trying with 250 components
Explained variance: 0.61
Trying with 300 components
Explained variance: 0.66
Trying with 350 components
Explained variance: 0.71
Trying with 400 components
Explained variance: 0.74
Trying with 450 components
Explained variance: 0.78
Trying with 500 components
Explained variance: 0.81
Trying with 550 components
Explained variance: 0.83
Trying with 600 components
Explained variance: 0.86
Trying with 650 components
Explained variance: 0.88
Trying with 700 components
Explained variance: 0.90
Trying with 750 components
Explained variance: 0.91
Trying with 800 components
Explained variance: 0.93
Trying with 850 components
Explained variance: 0.94
Trying with 900 components
Explained variance: 0.96
Optimal number of components: 900
```

In [21]:
```
# Create the best model with the optimal number of components
svd = TruncatedSVD(n_components=best_n_components)
normalizer = Normalizer(copy=False)
lsa = make_pipeline(svd, normalizer)
X_lsa = lsa.fit_transform(X)

print(f'Best model saved with {best_n_components} components')
```

```
Best model saved with 900 components
```

In [22]:
```
# Visualizae the important terms
terms = lsa.get_feature_names_out()

# Calculate average TF-IDF scores for each feature
average_tfidf_scores = X_lsa.mean(axis=0).tolist()
```

```
df = pd.DataFrame({'term': terms, 'average_tfidf': average_tfidf_scores})

# Sort by average TF-IDF scores in descending order
df_sorted = df.sort_values(by='average_tfidf', ascending=False)

# Select top N features with the highest average TF-IDF scores
top_n = 10
x = df_sorted['term'].head(top_n)
y = df_sorted['average_tfidf'].head(top_n)

# Create a bar plot using matplotlib
plt.figure(figsize=(30, 8))
plt.bar(x, y)
plt.xlabel('Terms')
plt.ylabel('Average TF-IDF Score')
plt.title('Bar Plot of Terms and Their Corresponding TF-IDF Scores')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```



K-means Clustering

In [23]:
```
max_clusters = len(X_lsa)
best_score = 100
best_model = None

for n_clusters in np.arange(50,max_clusters,50):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init='auto')
    cluster_labels = kmeans.fit_predict(X_lsa)
    calinski_harabasz = calinski_harabasz_score(X_lsa, cluster_labels)
    silhouette_avg = silhouette_score(X_lsa, cluster_labels)
    print(f'For n_clusters={n_clusters}, the average Calinski-Harabasz score is {ca
    if calinski_harabasz < best_score:
        print('Model saved as best')
        best_score = calinski_harabasz
        best_model = kmeans
print(f'Best Score: {best_score}, with # of clusters: {best_model.n_clusters}')

# Assign cluster labels
text_frame['cluster'] = best_model.labels_
```

```
For n_clusters=50, the average Calinski-Harabasz score is 5.74 and the Silhouette sc
ore of : 0.05
Model saved as best
For n_clusters=100, the average Calinski-Harabasz score is 3.43 and the Silhouette s
core of : 0.05
Model saved as best
For n_clusters=150, the average Calinski-Harabasz score is 2.99 and the Silhouette s
core of : 0.06
Model saved as best
For n_clusters=200, the average Calinski-Harabasz score is 2.72 and the Silhouette s
core of : 0.09
Model saved as best
For n_clusters=250, the average Calinski-Harabasz score is 2.51 and the Silhouette s
core of : 0.10
Model saved as best
For n_clusters=300, the average Calinski-Harabasz score is 2.41 and the Silhouette s
core of : 0.11
Model saved as best
For n_clusters=350, the average Calinski-Harabasz score is 2.38 and the Silhouette s
core of : 0.13
Model saved as best
For n_clusters=400, the average Calinski-Harabasz score is 2.34 and the Silhouette s
core of : 0.14
Model saved as best
For n_clusters=450, the average Calinski-Harabasz score is 2.29 and the Silhouette s
core of : 0.15
Model saved as best
For n_clusters=500, the average Calinski-Harabasz score is 2.27 and the Silhouette s
core of : 0.15
Model saved as best
For n_clusters=550, the average Calinski-Harabasz score is 2.25 and the Silhouette s
core of : 0.15
Model saved as best
For n_clusters=600, the average Calinski-Harabasz score is 2.26 and the Silhouette s
core of : 0.16
For n_clusters=650, the average Calinski-Harabasz score is 2.29 and the Silhouette s
core of : 0.16
For n_clusters=700, the average Calinski-Harabasz score is 2.36 and the Silhouette s
core of : 0.17
For n_clusters=750, the average Calinski-Harabasz score is 2.43 and the Silhouette s
core of : 0.17
For n_clusters=800, the average Calinski-Harabasz score is 2.57 and the Silhouette s
core of : 0.18
For n_clusters=850, the average Calinski-Harabasz score is 2.79 and the Silhouette s
core of : 0.18
For n_clusters=900, the average Calinski-Harabasz score is 3.11 and the Silhouette s
core of : 0.19
For n_clusters=950, the average Calinski-Harabasz score is 3.77 and the Silhouette s
core of : 0.19
For n_clusters=1000, the average Calinski-Harabasz score is 4.97 and the Silhouette
score of : 0.20
For n_clusters=1050, the average Calinski-Harabasz score is 6.94 and the Silhouette
score of : 0.18
For n_clusters=1100, the average Calinski-Harabasz score is 10.63 and the Silhouette
score of : 0.18
For n_clusters=1150, the average Calinski-Harabasz score is 19.70 and the Silhouette
```

```
score of : 0.17
For n_clusters=1200, the average Calinski-Harabasz score is 41.52 and the Silhouette
score of : 0.14
For n_clusters=1250, the average Calinski-Harabasz score is 124.65 and the Silhouett
e score of : 0.10
For n_clusters=1300, the average Calinski-Harabasz score is 119788150565327218622728
668971008.00 and the Silhouette score of : 0.04
Best Score: 2.251696852513715, with # of clusters: 550
```

Results

The resulting model can be used to gather test cases from contextual categories (names of the categories are irrelevant as long as they are effictive) and predict the category of a search term. Then the cosine similarity can rank the results from that category and return the top 5 cases for the user to review.

In [24]:
```python
# Match function
def step_match(input_text):
    input_vector = vectorizer.transform([input_text])
    input_lsa = lsa.transform(input_vector)
    similarities = cosine_similarity(input_lsa, X_lsa)
    best_cluster = best_model.predict(input_lsa)[0]
    cluster_data = text_frame[text_frame['cluster'] == best_cluster].copy()
    cluster_similarities = similarities[0][cluster_data.index]
    if cluster_similarities.max() < 0.05:
        return None
    cluster_data['similarity'] = cluster_similarities
    best_matching_cases = cluster_data.sort_values(by='similarity', ascending=False
    filtered_cases = data[data['Key'].isin(best_matching_cases['Key'].head(5))]
    return filtered_cases
```

Example Searches

In [25]:
```python
search_terms = ['Auto Level is not responding properly to the pitch sensor',
                'Mobius should load drill plans more quickly',
                'Drill plans checksum is not being checked properly']
for search in search_terms:
    print('\n', search)
    matching_cases = step_match(search)
    if matching_cases is not None:
        for case in matching_cases.iterrows():
            print(case[1]['Objective'])
    else:
        print('No matching cases returned for that search. Be more specific or add
```

Auto Level is not responding properly to the pitch sensor
If the pitch and roll sensors are updated on every sync the OU will freeze.Try to replicate this in lab by turning on ripple on these sensors and allocate the rig using the OU.Investigate if we can filter or limit the number of updates we send to the OU.This issue was reported from Aitik running 5.3 rev.6 but we can test in the latest version...
91000 Display does not boot up on the Rig and in the sim when trying to run without AD2 configured. It may be a configuration mismatch but it was tested without ADPE, then without ADPE and ARC, and then with no ARC configured with no change. It isn't clear what the mismatch might be. Config that loads: 91000-SYSCONF.TXTConfig with one change that doesn't load: 91000 Doesn't load SYSCONF.TXT
Machine type:N/A (all)Software version:5.6.0-alpha6 + ~50 change setsProblemThe OPS displays a ATA menu button on the setup screen. When the only option is ATA no menu buttons should be visible.Expected behaviorWhen only ATA is enabled (not ABC), the menu button on the setup screen should be hidden.How to reproduceTurn off ABCAllocate from OPSGo to setup screenResolutionWhat was done in the issue? Write it down before the issue is reviewed.
Discovered while testingADSRTE-2297&;Previously, Autolevel would hang and timeout when rig level exceeded both align limits and an autodelevel was attempted.&;This bug was fixed but now AutoLevel is ignoring the align limits and moving sending an equal all jacks up command.If a jack pulsar is not working and auto delevel is used an all jack command will leave one jack extended and retract all other jacks which will tilt the machine. In this case it should stop trying to give an all jack up command if it goes out of align limits. If it can;t get it into align limits it will time out and exit.&;What it should do:At the start of auto delevel if the rig is over an align limit it should rise the jack(s) to bring the machine down within the align limits then send an all jacks up command.At the start of auto delevel If the rig is over both align limits the machine should fix the angle that is the most over the limit until it is within align limits. Then it should fix the other angle until it is within the align limit. after both angles are within align limits it should give an all jacks up command.During an all jacks up command if one of the align limits are exceeded the all jacks up command should stop and&;it should rise the jack(s) to bring the machine within the align limits before going back to all jacks up command. If the align limit does not improve Autolevel will time out.&;
Logged in as SE or higher, the Auto level parameters, Settings - Setup - Parameters - Autolevel, are visible and editable on rigs without Auto Level configured.Attempting to start autolevel (auto button then toggle on the 2 axis stick) has no effect so it seems to be limited to GUI access.

Mobius should load drill plans more quickly
Multiple drill plans have been tested, but none of them will load from a USB to the Rig. The same drill plans can be loaded if they are sent to the CCI though the RRA or even if they are first loaded from the USB to the CCI using the Data screen.&;&;This drill plans will attempt to load, (Green circle spins) but will fail and RCS pops up to report (Failed Loading Drill Plan)&;&;Testing was done on multiple machines and found that it applies to both the lab and the rigs but not the sim. It was also tested in 5.4.017 and is not an issue in that version so this is new to 5.4.018.&;User files and TUM codes can still be loaded from the USB. Access Level Overrides can't be but that has been reported in a separate Jira&;ADSRTE-4222.
We have found every third time a drill plan is selected on onsite OPS there is an extended loading pop up and ends with a file transfer error. This was tried on 2 different ops stations using 91000 and 8488, as well as starting a sim using an onsite ops station. after the error pop up it will allow you to load a drill plan 2 times then give the same fault.(this has more to do with the timing between loading the files. If you load too quickly after spinner goes away it can be reproduced.)What was found:When a drill plan load is commanded there is a 5 sec delay before another drill

plan load command can be given. If the plan is loaded before the 5 sec expires then the spinner will stop and you can not load a drill plan until the 5 sec timer has expired.What was done:The change will be to cancel the timer when the load is complete. Edge case that will not be fixed at this time:On allocation from ops station if you try to load the drill plan within the 5 sec timer the plan will not load. You will then have to wait for the time out from the ops station. (This is not new to 4.20) when a user creates a drill plan sequence and drags one of the holes to change the order the following error appears:
The Mobius client can enter a state where the user is unable to publish drill plans. It looks almost as if the circles for publishing drill plans have been greyed out. It takes a number of reboots of the client to make the publish circles active again.Please see the video ""Investigation is still ongoing into what causes this.One of the times this happened the groups tab kept expanding and contracting in the drill plan menu and in the archive menu. Please see video "Group Dropdown Activates Multiple Times.MOV"

Drill plans checksum is not being checked properly
Multiple drill plans have been tested, but none of them will load from a USB to the Rig. The same drill plans can be loaded if they are sent to the CCI though the RRA or even if they are first loaded from the USB to the CCI using the Data screen.&;&;This drill plans will attempt to load, (Green circle spins) but will fail and RCS pops up to report (Failed Loading Drill Plan)&;&;Testing was done on multiple machines and found that it applies to both the lab and the rigs but not the sim. It was also tested in 5.4.017 and is not an issue in that version so this is new to 5.4.018.&;User files and TUM codes can still be loaded from the USB. Access Level Overrides can't be but that has been reported in a separate Jira&;ADSRTE-4222.
Test the links between the table clamp, rotary torque control and ARC options to make sure they activate as desired
When booting the offsite and onsite ops stations, RCS reports "Checksum Error in Config File," even though we get this pop-up there doesn't seem to be any actions or operations that we are unable to perform.
"Enable Log" checkbox should be checked
After attempting to 'Load' parameters from 'DISP' with no USB in slot, the event should display 'No card in slot' but it actually displays 'Checksum Error in Parameter File'.The 'Checksum Error in Parameter File' should only be displayed when the DATBASE.TXT is present and the checksum is incorrect.

Conclusion

The results of this project are extremely promising. The manual testing of matching cases has resulted in relevant cases every time I have attempted it. I don't have a good grasp on cases it might be missing, but the cases returned are obviously related to the intial search. The biggest benefit of this structure is that the result seems to be better the more context given as a search term. Specifically, it is valid to drop in the name of a stofware change request and expect to get back relevant cases. Since the tester's don't write the name of the software change requests, this can be automated to perform a search and return results before the testers need to spend any time or energy trying to determine the scope of the case on their own. One of the potential uses (beyond the scope of the project) is to feed these test cases into an AI LLM as context in order to ask the AI for help analyzing the test cases and even to suggest areas where the code might be undertested.