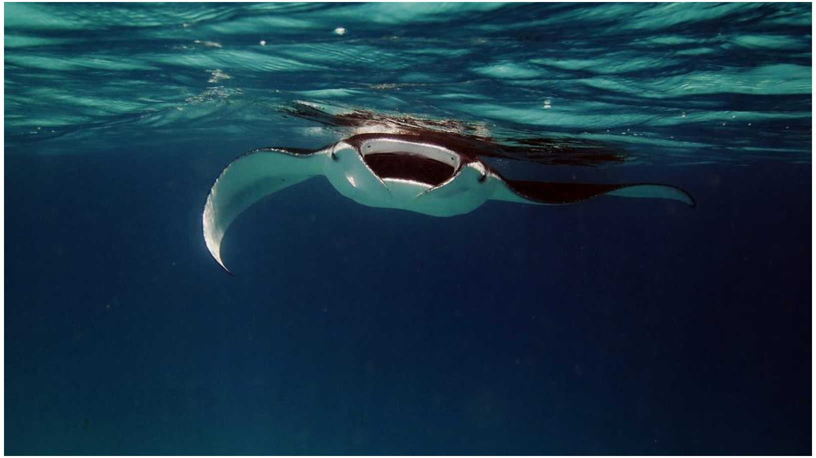
The General: NLP Predictions

Rachael Abram
Selam Tekie
Davis Thrailkill



4 Predictions Based on Text

1. Who’s at fault?
2. What claim group should it be routed to?
3. What is the severity type?
4. What is the loss cause?

Can you predict who is a fault for an accident?

Who is at fault?

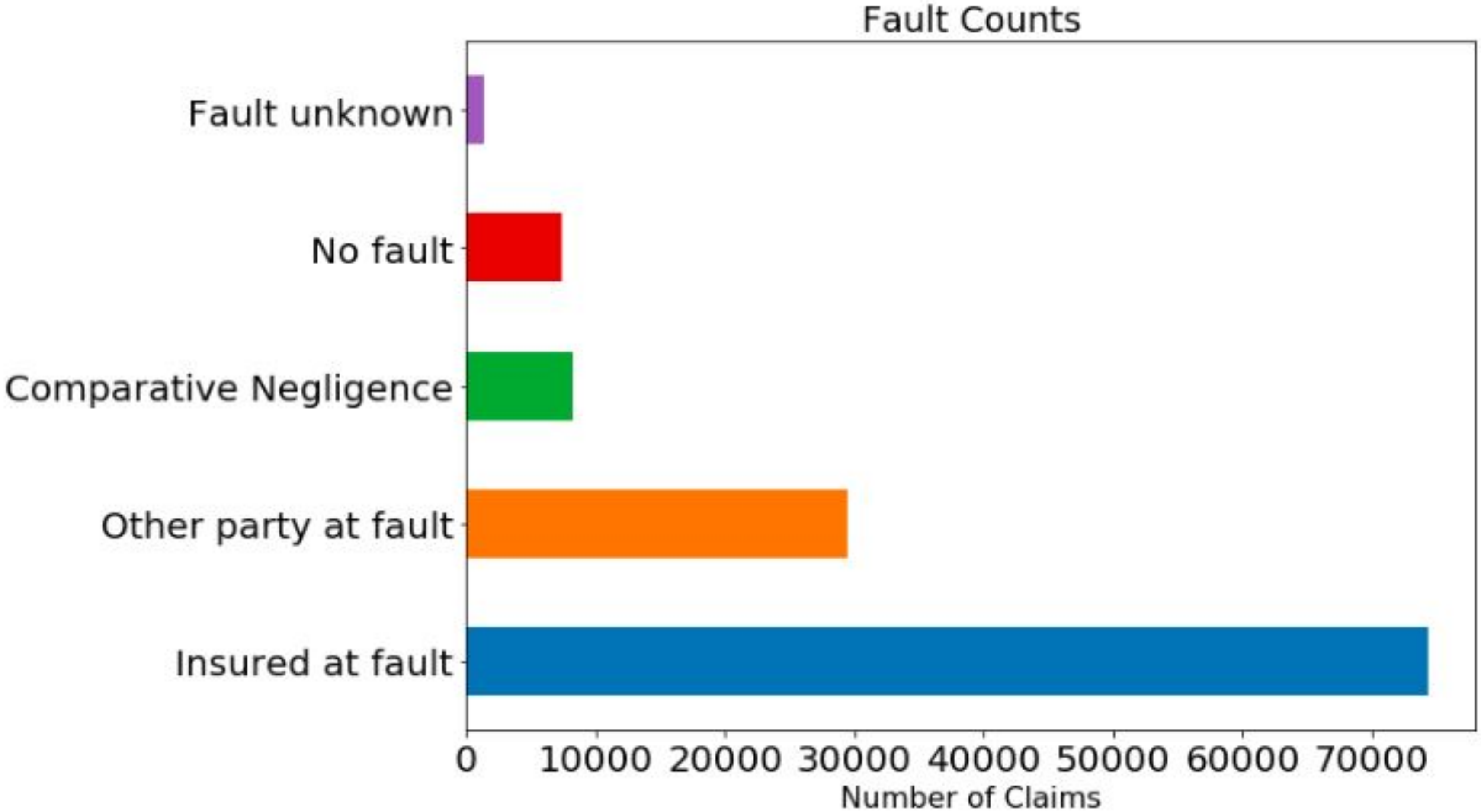
- Insured at fault
  - IV rear ended CV
- Other party at fault
  - CV rear ended IV
- Comparative negligence
  - iv ran a red light causing iv to strike cv causing cv to strike a pole
- No fault
  - IV struck a deer
- Fault unknown
  - iv was making a left turn at an intersection and iv struck a pedestrian iv states he crossed the road behind a truck and the pedestrian walked into iv

Why does it matter who's at fault?

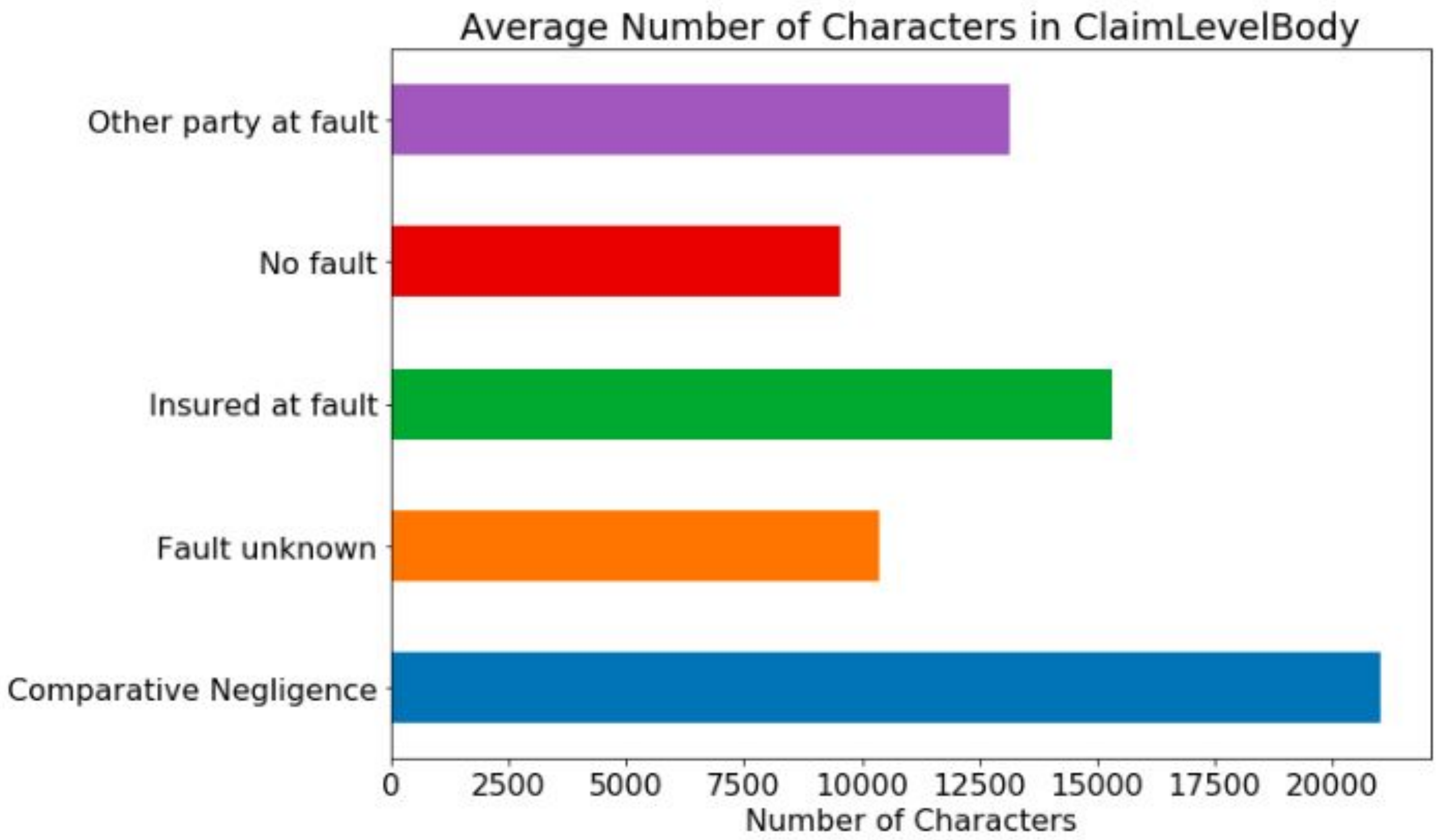
Step 1: Cleaning / Pre-Processing

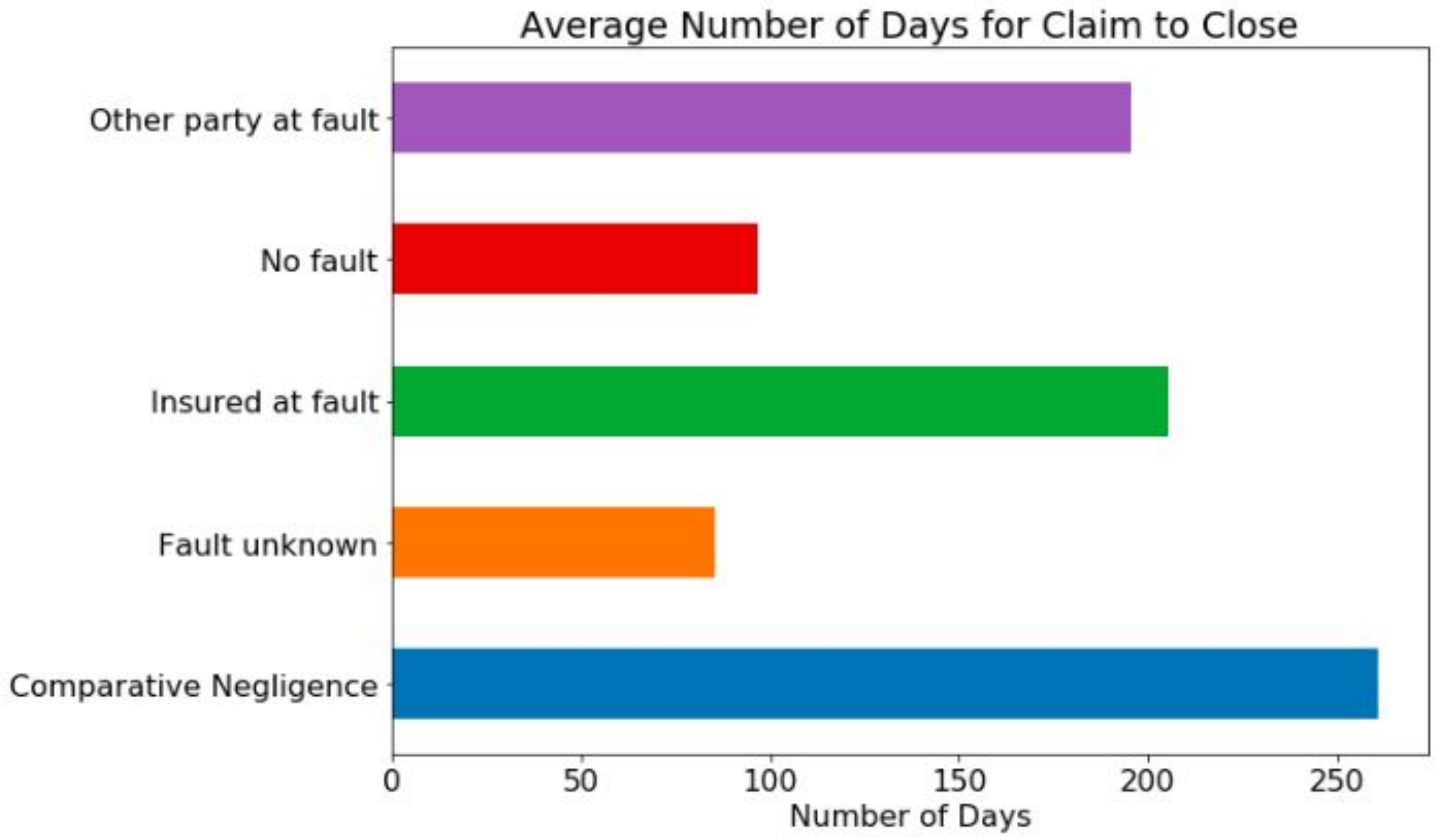
- Claim-level question → Drop multiple exposures to avoid overfitting
    - Done on CCCreateTime since it’s unique
- Fix characters
    - *
    - \r
    - \n
    - re
- Remove stop words and punctuations
- Convert to lowercase

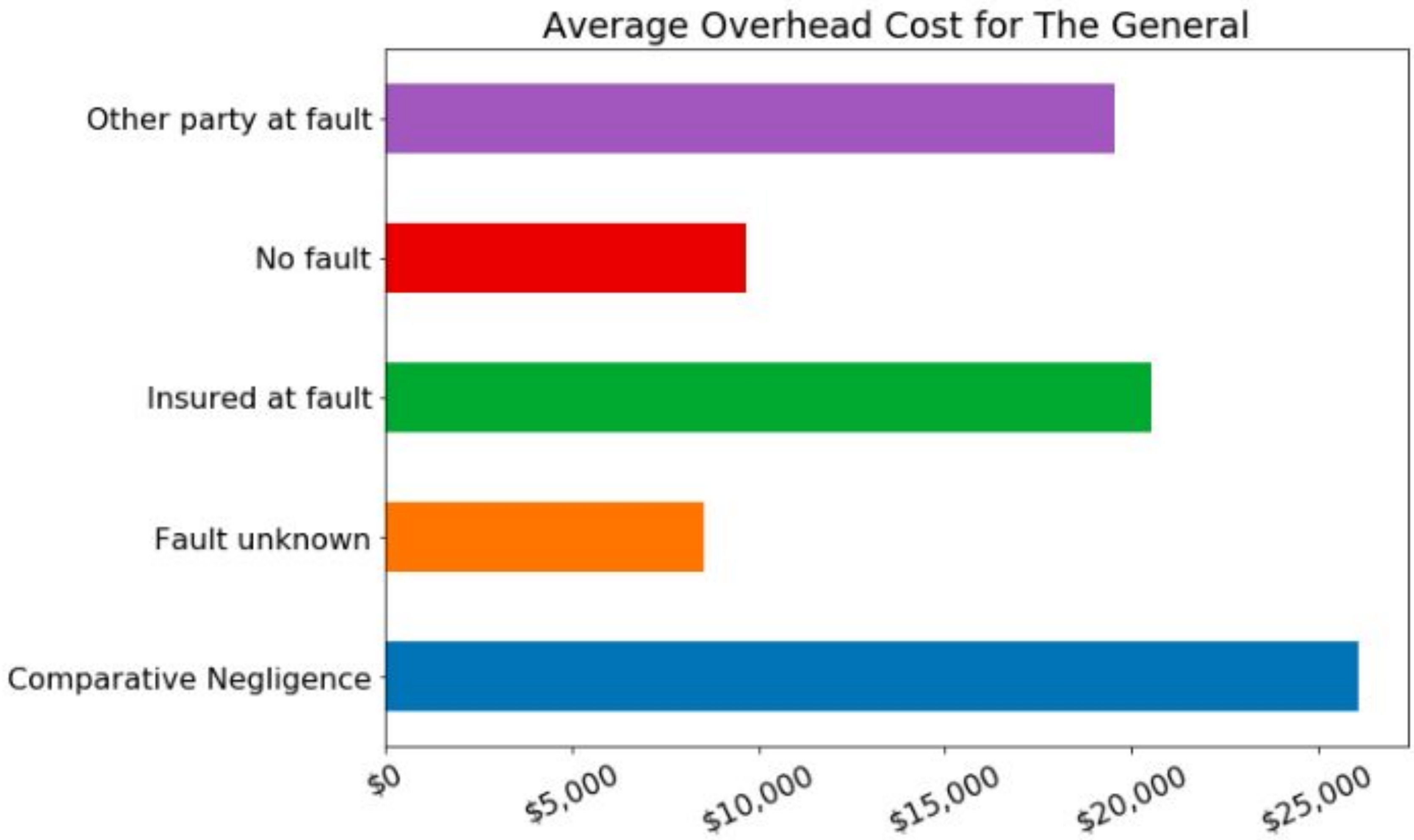
Step 2: EDA



Fault Counts
Fault unknown
No fault
Comparative Negligence
Other party at fault
Insured at fault
0
10000
20000
30000
40000
50000
60000
70000
Number of Claims


Average Number of Characters in ClaimLevelBody
Other party at fault
No fault
Insured at fault
Fault unknown
Comparative Negligence
0
2500
5000
7500
10000
12500
15000
17500
20000
Number of Characters


Average Number of Days for Claim to Close
Other party at fault
No fault
Insured at fault
Fault unknown
Comparative Negligence
0
50
100
150
200
250
Number of Days


Average Overhead Cost for The General
Other party at fault
No fault
Insured at fault
Fault unknown
Comparative Negligence
$0
$5,000
$10,000
$15,000
$20,000
$25,000

Step 3: Machine Learning

- Which column of text to use as features?
  - ClaimLevelBody vs. Accident Description vs. Damage Description
- Which vectorizer to use?
  - CountVectorizer vs. TF-IDF
- Which classifier?
  - Naive Bayes vs. SGD

```python
vect = CountVectorizer(tokenizer=tokenizer, ngram_range=(1,2))
clf = SGDClassifier(loss='log', max_iter=100, tol=1e-6, random_state=42)
```

```
Accuracy Score: 0.8460943542150039
                          precision    recall  f1-score   support

Comparative Negligence       0.57      0.20      0.30       911
         Fault unknown       0.00      0.00      0.00         3
      Insured at fault       0.87      0.96      0.91      9210
              No fault       0.47      0.29      0.36       716
  Other party at fault       0.84      0.83      0.84      3383

           avg / total       0.83      0.85      0.83     14223
```

Key Features

Comparative Negligence: speed majority comp negligence lookout compneg decision comparative comp neg neg

Fault unknown: veh ticket report alexandria pc you pc to lessner insd or

Insured at fault: insd at from clmt iv ran liability adverse control single fol iv when iv into cv iv rear

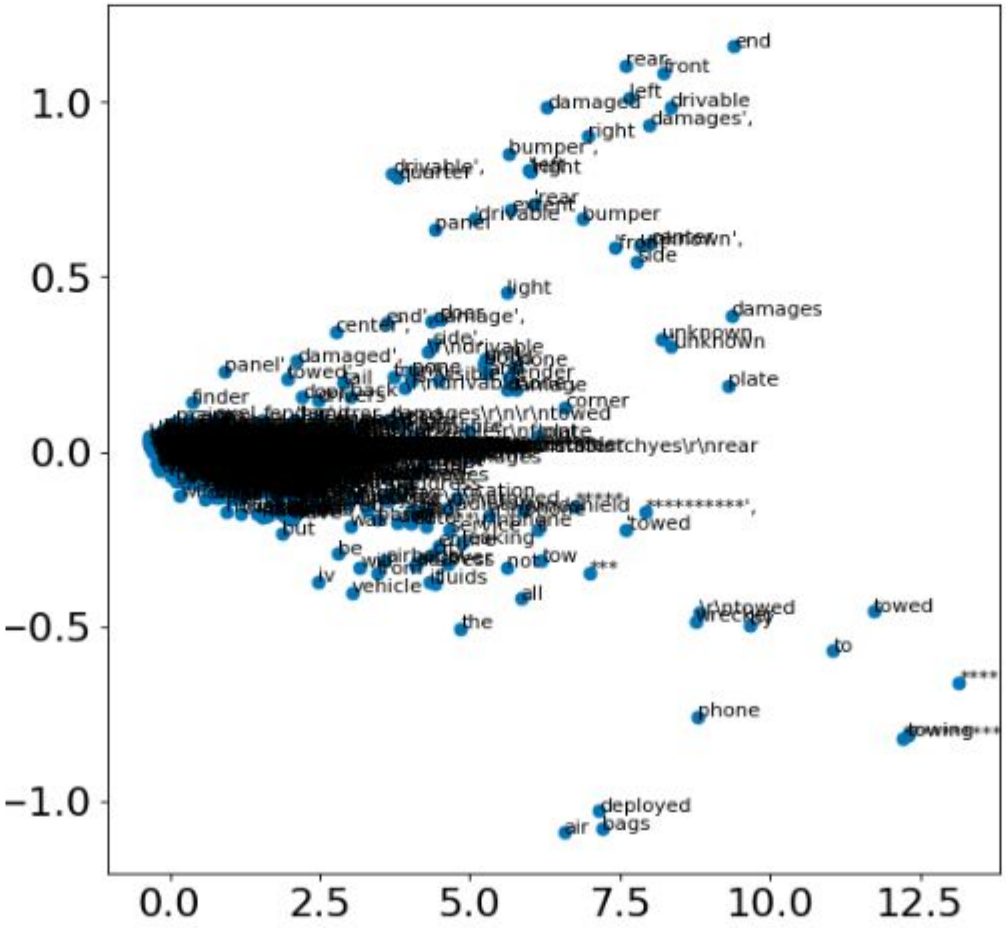
No fault: a deer jetta lor shooting suspect markus comp theft shot deer

Other party at fault: liability denial rear endediv umpd ended iv accepted liability fol cv when cv cv rear struck iv umbi

Word2Vec

```
model.wv.most_similar(positive='vehicle')
```

```
[('had', 0.9995748400688171),
 ('car', 0.999489963054657),
 ('insured', 0.9994364976882935),
 ('pole', 0.9993816614151001),
 ('iv', 0.999369323253316),
 ('cut', 0.999318182468414З),
 ('quitman', 0.9993139505386353),
 ('be', 0.9992308616638184),
 ('services', 0.9992300271987915),
 ('he', 0.9992241859436035)]
```



Can you predict what group the claim should be routed to?

Cleveland Field Ops	9030
PIP	8537
Tampa Field Ops	5143
Phoenix Casualty Ops	5130
Nashville Field Ops	4910
Nashville Casualty Ops	4213
Atlanta Casualty Ops	3568
Large Loss 2	3262
Cleveland Casualty Ops	3003
CCU	3002
NARBI	2779
Tampa Casualty Ops	2528
Atlanta Field Ops	2513
Large Loss 1	2396
Phoenix West Field Ops	2244
Albany Casualty Ops	1969
Total Loss	1680
Phoenix East Field Ops	1634
Claims Overflow	1530
Fast Track	1390
DMA Vendor	1121
Albany Field Ops	1095
Inbound Subrogation	887
Recoveries	508
Inactive	449
Specialty	389
Large Loss 3	319
Non Claims	305
Central SIU	264
Claims QA	195
West Coast SIU	134
TAG	39
Premium Fraud SIU	8
Field Ops 2 Executive	7
CCU Executive	1
Dispatch	1
Admin	1

Field Ops	26587
Casualty Ops	20411
PIP	8537
Loss	7657
CCU	3002
NARBI	2779
Claims Overflow	1530
Fast Track	1390
DMA Vendor	1121
Inbound Subrogation	887
Recoveries	508
Inactive	449
SIU	406
Specialty	389
Non Claims	305
Claims QA	195
TAG	39
Dispatch	1
Admin	1
CCU Executive	1

Field Ops	26587
Casualty Ops	20411
PIP	8537
Loss	7657
CCU	3002
NARBI	2779

```
vect = CountVectorizer(tokenizer=tokenizer, ngram_range=(1,2))
clf = SGDClassifier(loss='log', max_iter=100, tol=1e-6, random_state=42)
```

```
Accuracy Score: 0.8117915824619981
```

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
CCU	0.84	0.67	0.74	1012
Casualty Ops	0.86	0.86	0.86	6702
Field Ops	0.81	0.86	0.84	8782
Loss	0.73	0.80	0.76	2539
NARBI	0.82	0.56	0.67	905
PIP	0.76	0.68	0.72	2822
avg / total	0.81	0.81	0.81	22762

Key Features

CCU: ccu sol lit emailed plntf served pl pc suit revd

Casualty Ops: complete bie rept change of bie pc to pc attny ird rec clamnt eh

Field Ops: flags lm to ebi to ih closing ih called the for cb closing file expo

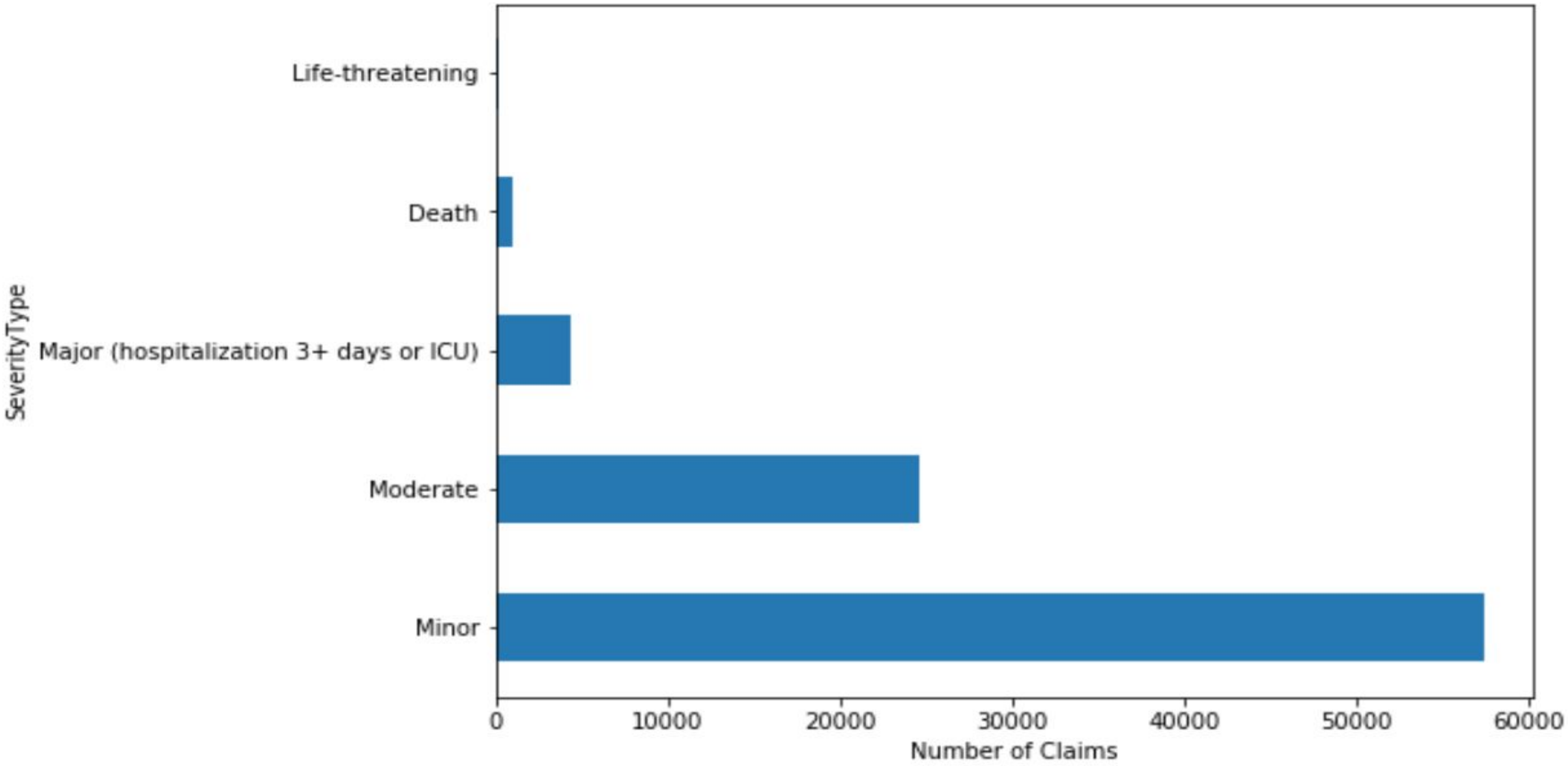
Loss: atc atty from atty i got dismissal ob call recommendations i called rcvd the only executed

NARBI: thanked wclmnt was able cnt placed phone call to lmtcb placed phone pat lmfcb

PIP: claim to medpay transfer transferring dilley aao oc ime pip sup to pip

Can you predict the severity type?

Severity Type



TFIDF & Multinomial Naive Bayes

```
]: tfidf = TfidfVectorizer(stop_words=stop_words, ngram_range=(2,3))

   tfidf_train = tfidf.fit_transform(X_train.values)
   tfidf_test = tfidf.transform(X_test.values)
```

```
]: nb_classifier.fit(tfidf_train, y_train)
   pred = nb_classifier.predict(tfidf_test)

   score = metrics.accuracy_score(y_test, pred)
   print(score)
```

```
0.6229481815255874
```

Stochastic Gradient Descent Classifier

```
X = model['ClaimLevelBody'].str.strip()
y = model['SeverityTypeName']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 42)
```

```
def tokenizer(text):
    return re.findall(r'[a-z0-9]+', text.lower())

vect = CountVectorizer(tokenizer=tokenizer)
clf = SGDClassifier(loss='log', max_iter=100, tol=1e-6, random_state=42)
```

```
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
```

```
clf.fit(X_train_tfidf, y_train)
y_pred = clf.predict(X_test_tfidf)
np.mean(y_test==y_pred)
```

```
0.6740490754411869
```

Stochastic Gradient Descent Classifier

Accuracy Score: 0.6740490754411869

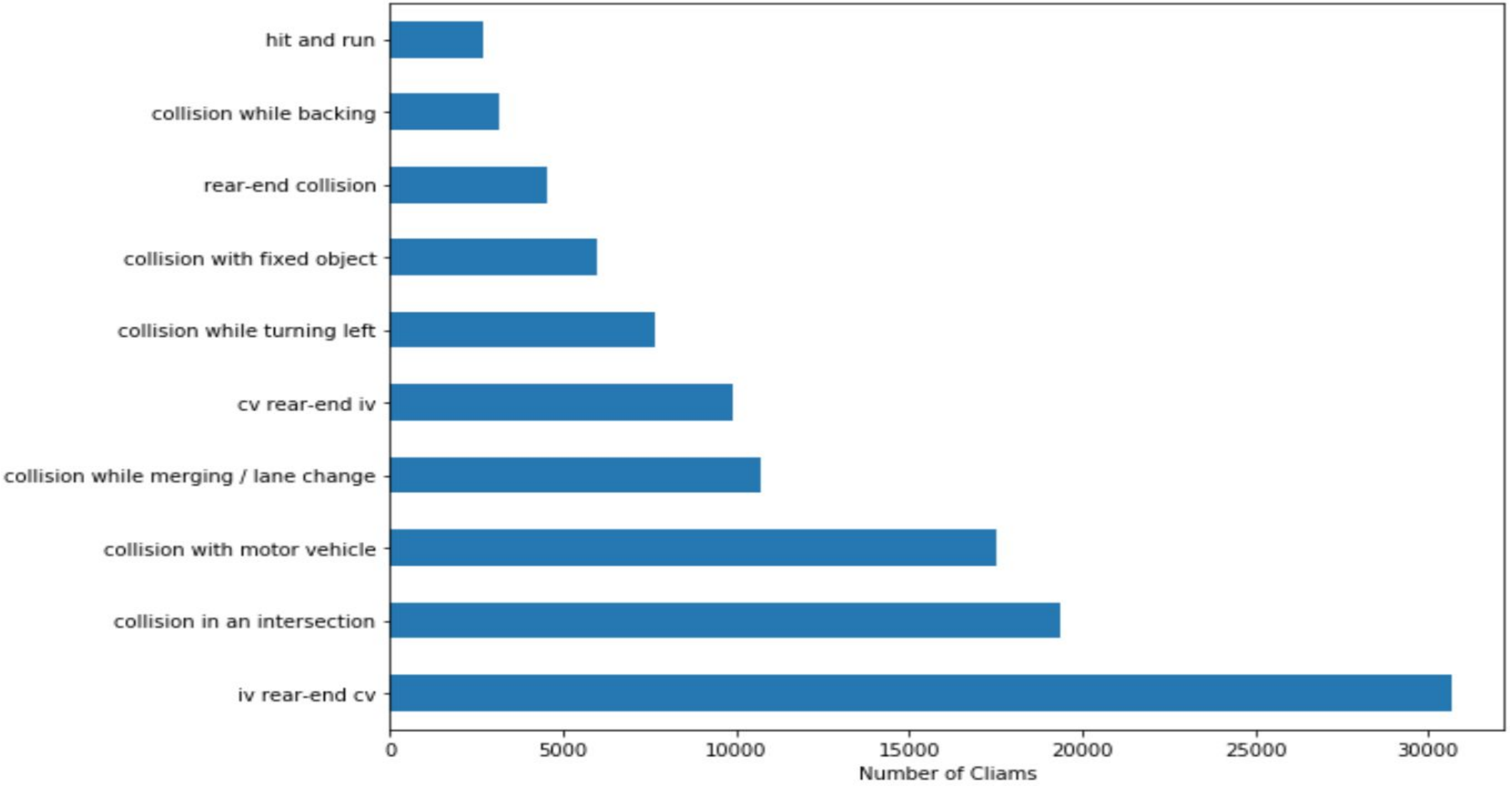
	precision	recall	f1-score	support
Death	1.00	0.04	0.08	136
Life-threatening	0.00	0.00	0.00	34
Major (hospitalization 3+ days or ICU)	0.61	0.17	0.27	716
Minor	0.69	0.98	0.81	9233
Moderate	0.48	0.11	0.18	4104
avg / total	0.63	0.67	0.59	14223

In [147]:
```
print_top10(tfidf, clf, class_labels)
```

```
Death: wrongful killed news died funeral fatal death deceased fatality estate
Life-threatening: torrie ped lybarger life montgomery fx tawanda icu coma pedestrian
Major (hospitalization 3+ days or ICU): ribs excess hospitalized fractured uim broken hospital fx surgery sir
Minor: minor atc tol attry rear sore ha lm claimant clamnt
Moderate: head concussion office used wrist best purposes law er attorney
```

Can you predict the loss cause?

Loss Cause Types



Preparing the Model

In [253]:
```
loss_cause_df.head()
```

Out[253]:

	CCCreateTime	AccidentDescription	DamageDescription	LossCauseName	LossCauseLabel	CombinedDescription
0	2015-03-12 09:05:17.910	the insured was test driving a vehicle the ov...	left side damages towed by unknown none front...	collision with motor vehicle	3	the insured was test driving a vehicle the ov...
1	2015-03-12 11:46:23.159	the iv was driving down the road when the ov i...	front right headlight front side of bumper dr...	collision while merging / lane change	4	the iv was driving down the road when the ov i...
2	2015-03-12 13:12:35.444	insured was stopped at the stop light when cv ...	unknown damages rear bumper trunk right rear ...	rear-end collision	8	insured was stopped at the stop light when cv ...
3	2015-03-12 13:12:35.444	insured was stopped at the stop light when cv ...	unknown damages rear bumper trunk right rear ...	rear-end collision	8	insured was stopped at the stop light when cv ...
4	2015-03-12 13:12:35.444	insured was stopped at the stop light when cv ...	unknown damages rear bumper trunk right rear ...	rear-end collision	8	insured was stopped at the stop light when cv ...

Combined Description with CountVectorizer and Multinomial Naive Bayes

In [254]:
```
X = loss_cause_df['CombinedDescription']
y = loss_cause_df['LossCauseLabel']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [255]:
```
cv = CountVectorizer(stop_words=stop_words)

count_train = cv.fit_transform(X_train.values)
count_test = cv.transform(X_test.values)
```

In [256]:
```
nb_classifier = MultinomialNB()

nb_classifier.fit(count_train, y_train)
pred = nb_classifier.predict(count_test)

score = metrics.accuracy_score(y_test, pred)
print(score)
```

```
0.6681948525507133
```

Combined Description with TFIDF and Multinomial Naive Bayes

In [257]:
```
tfidf = TfidfVectorizer(stop_words=stop_words, ngram_range=(2,3))

tfidf_train = tfidf.fit_transform(X_train.values)
tfidf_test = tfidf.transform(X_test.values)
```

In [258]:
```
nb_classifier.fit(tfidf_train, y_train)
pred = nb_classifier.predict(tfidf_test)

score = metrics.accuracy_score(y_test, pred)
print(score)
```

```
0.7717987750911601
```

Accident Description with CountVectorizer and Multinomial Naive Bayes

In [151]:
```
X = loss_cause_df['AccidentDescription']
y = loss_cause_df['LossCauseLabel']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [152]:
```
cv = CountVectorizer(stop_words=stop_words)

count_train = cv.fit_transform(X_train.values)
count_test = cv.transform(X_test.values)
```

In [153]:
```
nb_classifier = MultinomialNB()

nb_classifier.fit(count_train, y_train)
pred = nb_classifier.predict(count_test)

score = metrics.accuracy_score(y_test, pred)
print(score)
```

```
0.6300138067759408
```

Accident Description with TFIDF and Multinomial Naive Bayes

```
In [154]: tfidf = TfidfVectorizer(stop_words=stop_words, ngram_range=(2,3))

          tfidf_train = tfidf.fit_transform(X_train.values)
          tfidf_test = tfidf.transform(X_test.values)
```

```
In [155]: nb_classifier.fit(tfidf_train, y_train)
          pred = nb_classifier.predict(tfidf_test)

          score = metrics.accuracy_score(y_test, pred)
          print(score)

          0.7889864410379863
```

Next Steps

- Word2Vec → Pre-Trained
- Acronym Substitution
- 2+ columns of text to predict
- Focus on Comparative Negligence

Questions?