It's time to build a model

It's time to build a model

- Always a good approach to start with a very simple model
- Gives a sense of how challenging the problem is
- Many more things can go wrong in complex models
- How much signal can we pull out using basic methods?

It's time to build a model

- Train basic model on numeric data only
  - Want to go from raw data to predictions quickly
- Multi-class logistic regression
  - Train classifier on each label separately and use those to predict
- Format predictions and save to csv
- Compute log loss score

Splitting the multi-class dataset

- Recall: Train-test split
  - Will not work here
  - May end up with labels in test set that never appear in training set
- Solution: StratifiedShuffleSplit
  - Only works with a single target variable
  - We have many target variables
  - multilabel_train_test_split()

Splitting the data

```
In  [1]: data_to_train = df[NUMERIC_COLUMNS].fillna(-1000)

In  [2]: labels_to_use = pd.get_dummies(df[LABELS])

In  [3]: X_train, X_test, y_train, y_test = multilabel_train_test_split(
                                            data_to_train, labels_to_use,
                                            size=0.2, seed=123)
```

Training the model

```
In [4]: from sklearn.linear_model import LogisticRegression

In [5]: from sklearn.multiclass import OneVsRestClassifier

In [6]: clf = OneVsRestClassifier(LogisticRegression())

In [7]: clf.fit(X_train, y_train)
```

- OneVsRestClassifier:
  - Treats each column of y independently
  - Fits a separate classifier for each of the columns

https://github.com/drivendataorg/box-plots-sklearn/blob/master/src/data/multilabel.py

Let's practice!

Making predictions

Predicting on holdout data

```
In [1]: holdout = pd.read_csv('HoldoutData.csv', index_col=0)

In [2]: holdout = holdout[NUMERIC_COLUMNS].fillna(-1000)

In [3]: predictions = clf.predict_proba(holdout)
```

- If .predict() was used instead:
  - Output would be 0 or 1
  - Log loss penalizes being confident and wrong
  - Worse performance compared to .predict_proba()

Submitting your predictions as a csv

	Function__Aides Compensation	Function__Career & Academic Counseling	Function__Communications	...	Use__O&M	Use__Pupil Services & Enrichment	Use__Untracked Budget Set-Aside
180042	0.027027	0.027027	0.027027	...	0.125	0.125	0.125
28872	0.027027	0.027027	0.027027	...	0.125	0.125	0.125
186915	0.027027	0.027027	0.027027	...	0.125	0.125	0.125
412396	0.027027	0.027027	0.027027	...	0.125	0.125	0.125
427740	0.027027	0.027027	0.027027	...	0.125	0.125	0.125

- All formatting can be done with the pandas to_csv function

Format and submit predictions

```
In [4]: prediction_df = pd.DataFrame(columns=pd.get_dummies(df[LABELS],
   ...:                                   prefix_sep='__').columns,
   ...:                                   index=holdout.index,
   ...:                                   data=predictions)

In [5]: prediction_df.to_csv('predictions.csv')

In [6]: score = score_submission(pred_path='predictions.csv')
```

DrivenData leaderboard

User or team	Public	Private	Timestamp	Trend	# Entries
quocnle	0.3665	0.3650	Jan. 6, 2015, 12:27 a.m.		96
Abhishek	0.4409	0.4388	Jan. 6, 2015, 4:09 p.m.		71
giba	0.4551	0.4534	Jan. 5, 2015, 4:52 p.m.		34
trev	0.5054	0.5001	Jan. 3, 2015, 2 a.m.		23
Kappa	0.5228	0.5195	Jan. 6, 2015, 11:46 p.m.		17
bamine	0.5344	0.5298	Dec. 12, 2014, 12:52 a.m.		39
futuristic reality	0.5512	0.5477	Nov. 24, 2014, 8:54 a.m.		22
JesseBuesking	0.5584	0.5556	Jan. 6, 2015, 4:51 p.m.		15
mkrump	0.5817	0.5769	Jan. 3, 2015, 5:12 p.m.		57
joel314	0.5806	0.5772	Dec. 10, 2014, 4:41 p.m.		63

Let's practice!

A very brief introduction to NLP

A very brief introduction to NLP

- Data for NLP:
  - Text, documents, speech, …
- Tokenization
  - Splitting a string into segments
  - Store segments as list
- Example: ‘Natural Language Processing’
  - —> [‘Natural’, ‘Language’, ‘Processing’]

Tokens and token patterns

- Tokenize on whitespace

PETRO-VEND FUEL AND FLUIDS

PETRO-VEND | FUEL | AND | FLUIDS

- Tokenize on whitespace *and* punctuation

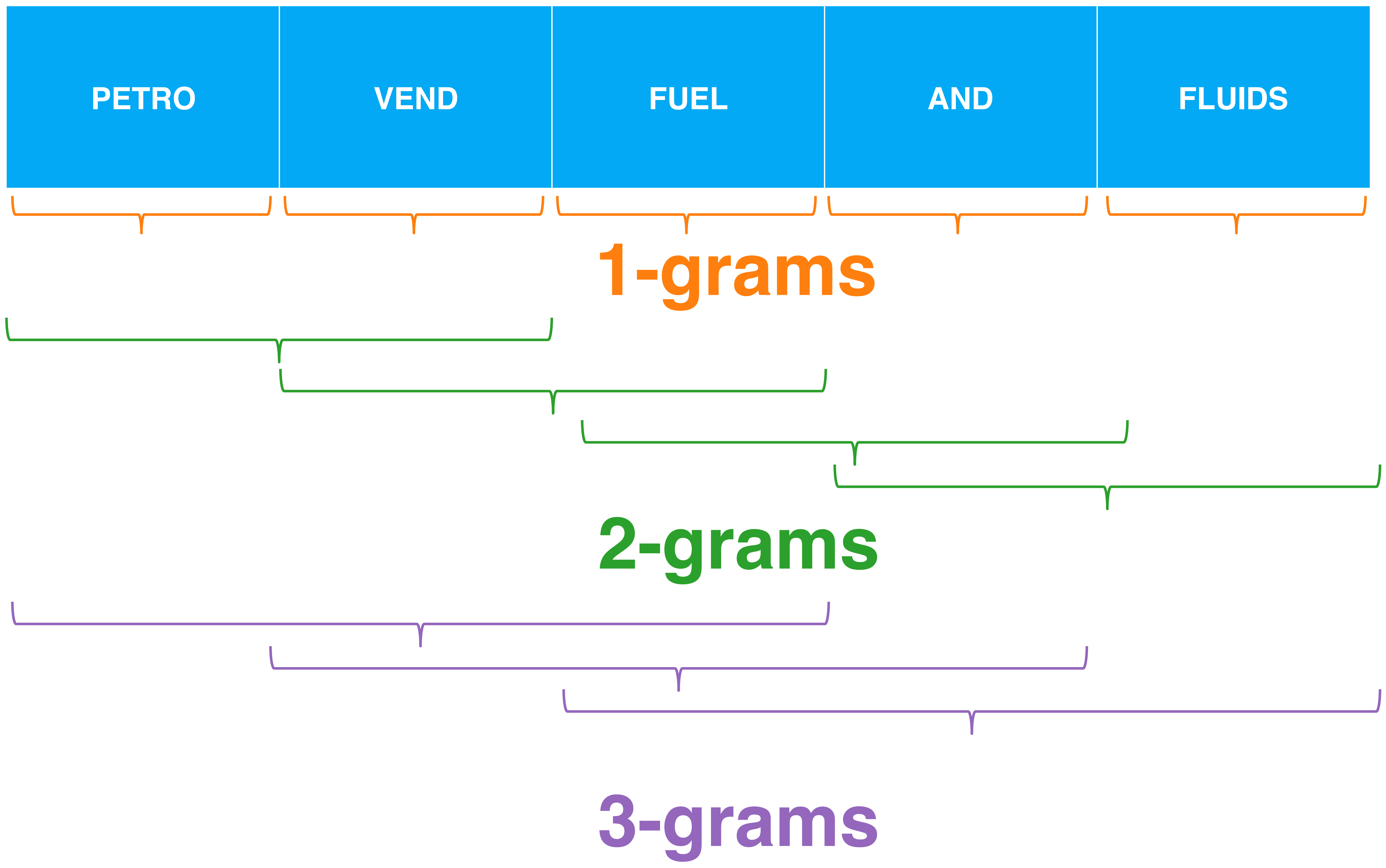
PETRO-VEND FUEL AND FLUIDS

PETRO | VEND | FUEL | AND | FLUIDS

Bag of words representation

- Count the number of times a particular token appears
- “Bag of words”
  - Count the number of times a word was pulled out of the bag
- This approach discards information about word order
  - “Red, not blue” is the same as “blue, not red”

1-gram, 2-gram, ..., n-gram



Let's practice!

Representing text numerically

Representing text numerically

- Bag-of-words
  - Simple way to represent text in machine learning
  - Discards information about grammar and word order
  - Computes frequency of occurrence

Scikit-learn tools for bag-of-words

- CountVectorizer()
  - Tokenizes all the strings
  - Builds a 'vocabulary'
  - Counts the occurrences of each token in the vocabulary

Using CountVectorizer() on column of main dataset

```
In [1]: from sklearn.feature_extraction.text import CountVectorizer

In [2]: TOKENS_BASIC = '\\S+(?=\\s+)'

In [3]: df.Program_Description.fillna('', inplace=True)

In [4]: vec_basic = CountVectorizer(token_pattern=TOKENS_BASIC)
```

Using CountVectorizer() on column of main dataset

```
In [5]: vec_basic.fit(df.Program_Description)
Out[5]:
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=1,
        ngram_range=(1, 1), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='\\S+(?=\\s+)',
        tokenizer=None, vocabulary=None)

In [6]: msg = 'There are {} tokens in Program_Description if tokens are
any non-whitespace'

In [7]: print(msg.format(len(vec_basic.get_feature_names())))
There are 157 tokens in Program_Description if tokens are any non-
whitespace
```

Let’s practice!