Applying Machine Learning to Storm Data Streams

Swetha Kolalapudi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

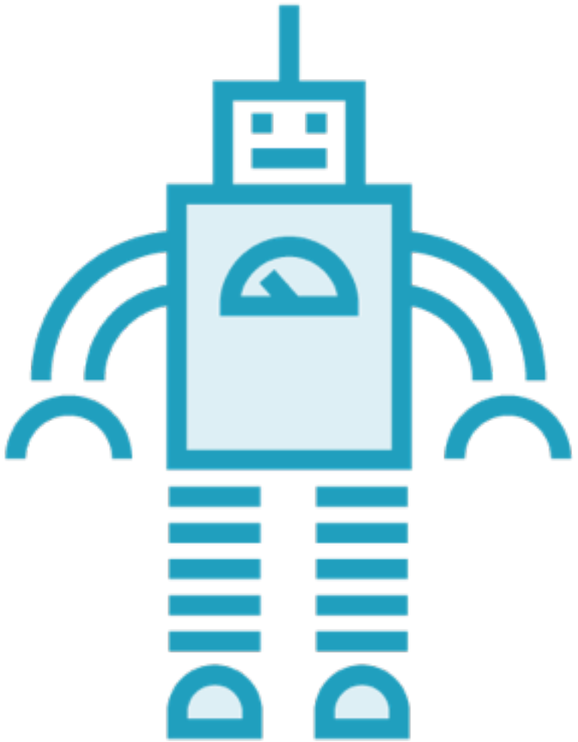
Overview

Understand the machine learning workflow i.e. training, testing

Integrate Python with Storm

Train a sentiment analysis model offline in Python

Use the model to predict sentiment in real-time with Storm



Machine Learning Problems

Current sentiment on Twitter for a brand

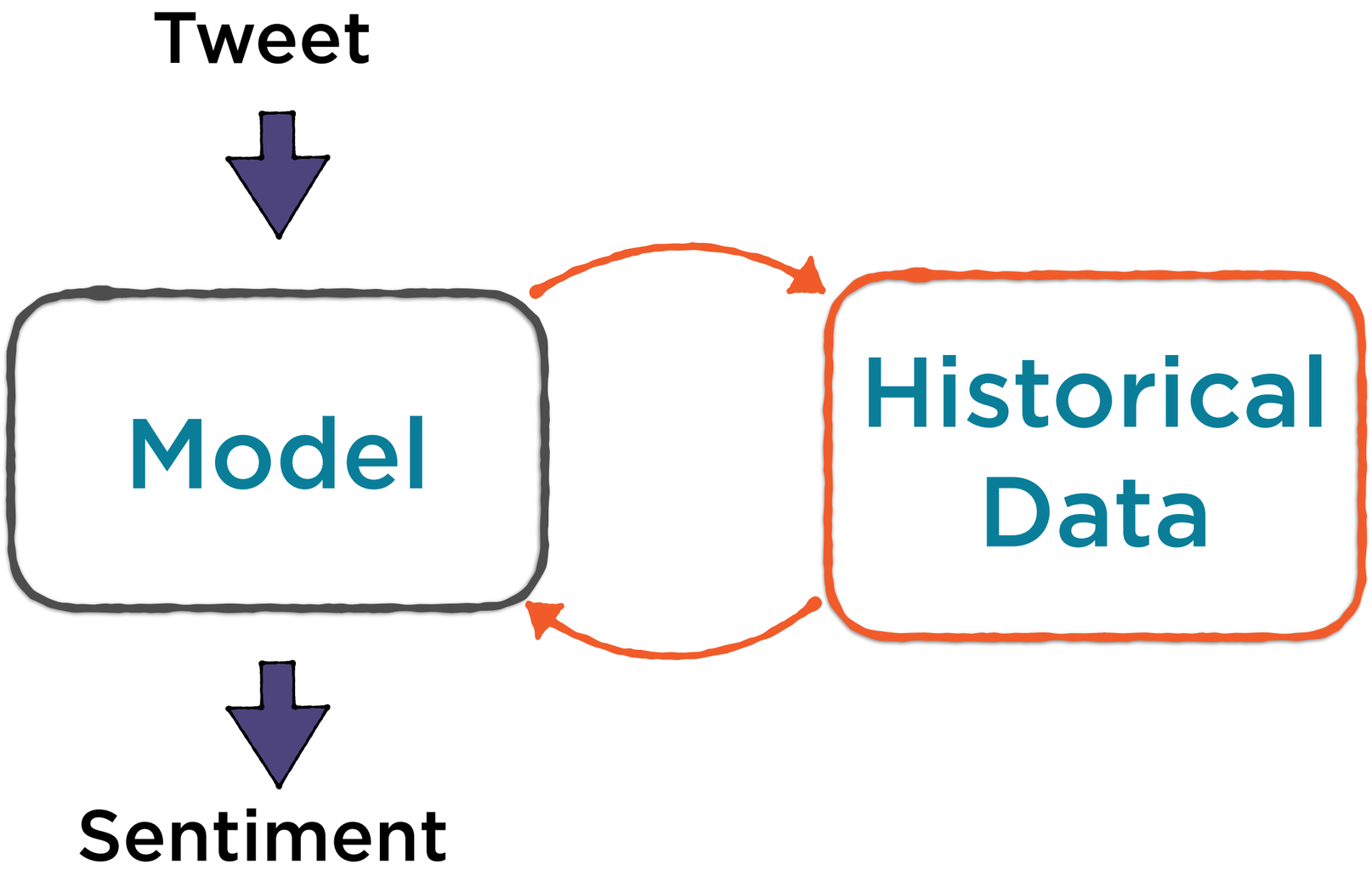
Projected returns for a security at the end of the day

Product recommendations for a user browsing the site

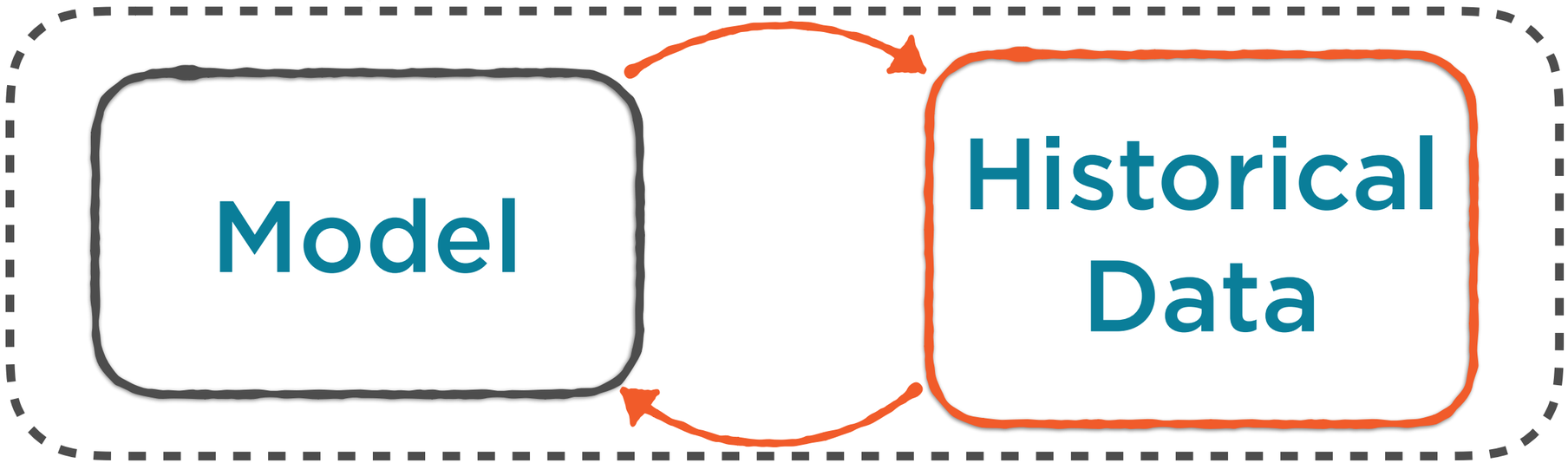
Expected time of arrival in a navigation app

Need real-time results

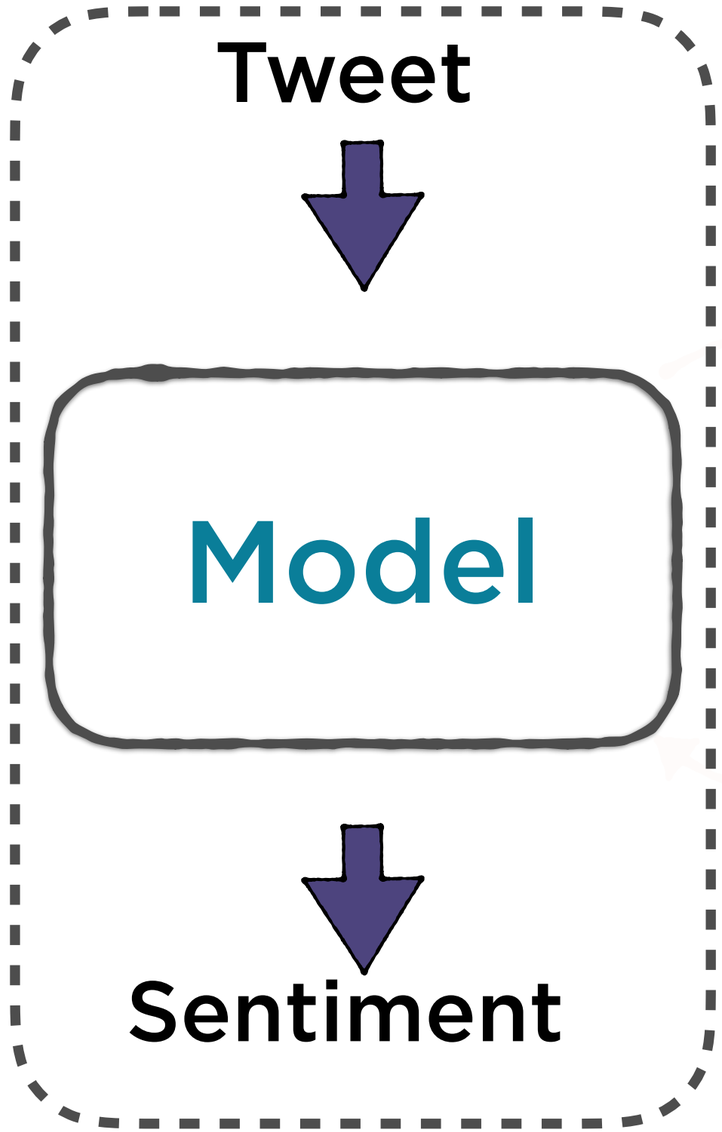
Machine Learning Process



Machine Learning Process



Machine Learning Process



Test Phase

Historical Data

Machine Learning Process

Training Phase

Train a model using historical data

Test Phase

Apply the model on new data

Training Phase

2 ways - offline and online

Offline

- Use all the training data at once

Online

- Use one data point at a time

Training Phase

Offline Learning

Use standard algorithms

- Naive Bayes, Support Vector Machines, Linear Regression

Evaluate the model against new data

Batch processing task

Offline Learning

Test Phase

Apply the model on incoming data

Real-time processing task

Machine Learning Process

Training Phase

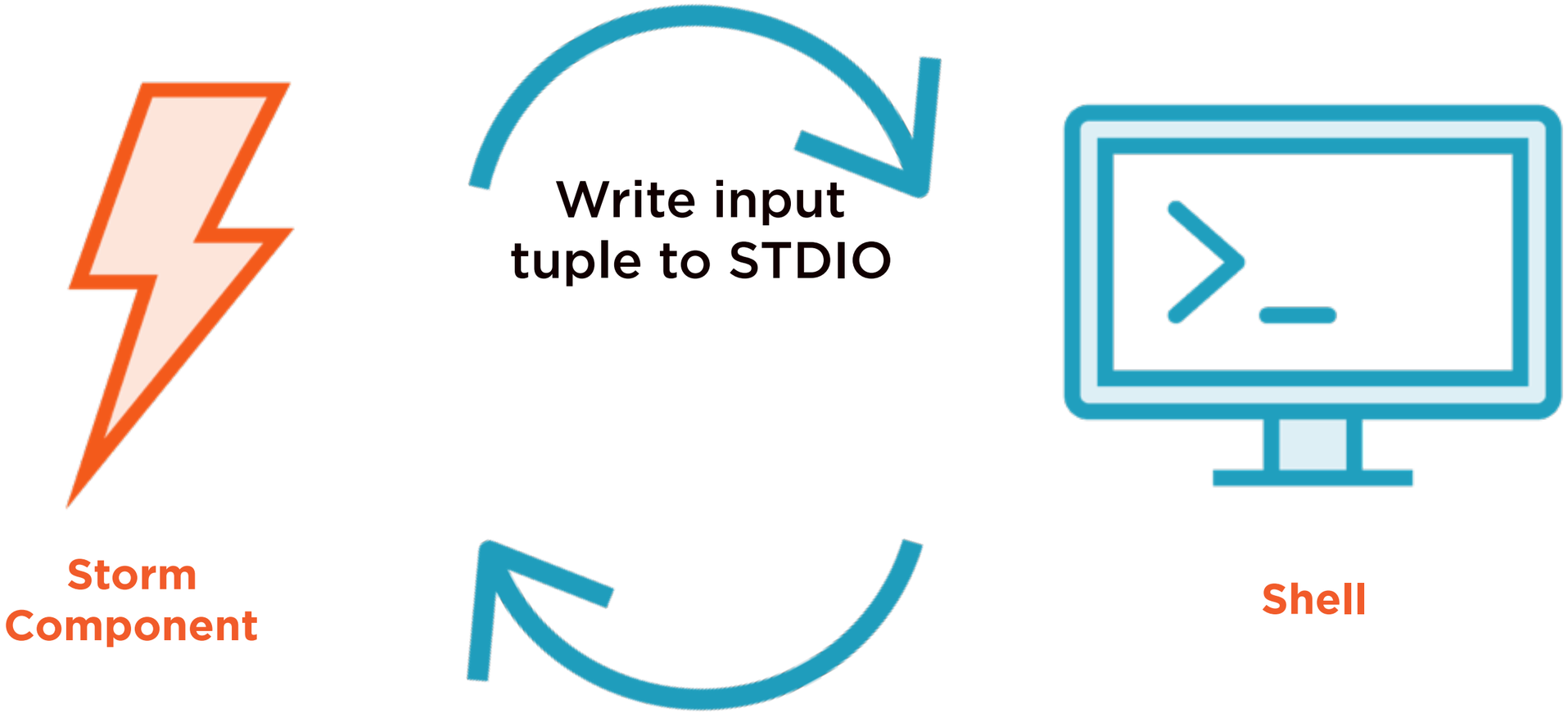
Use built-in libraries from Python, R etc

Test Phase

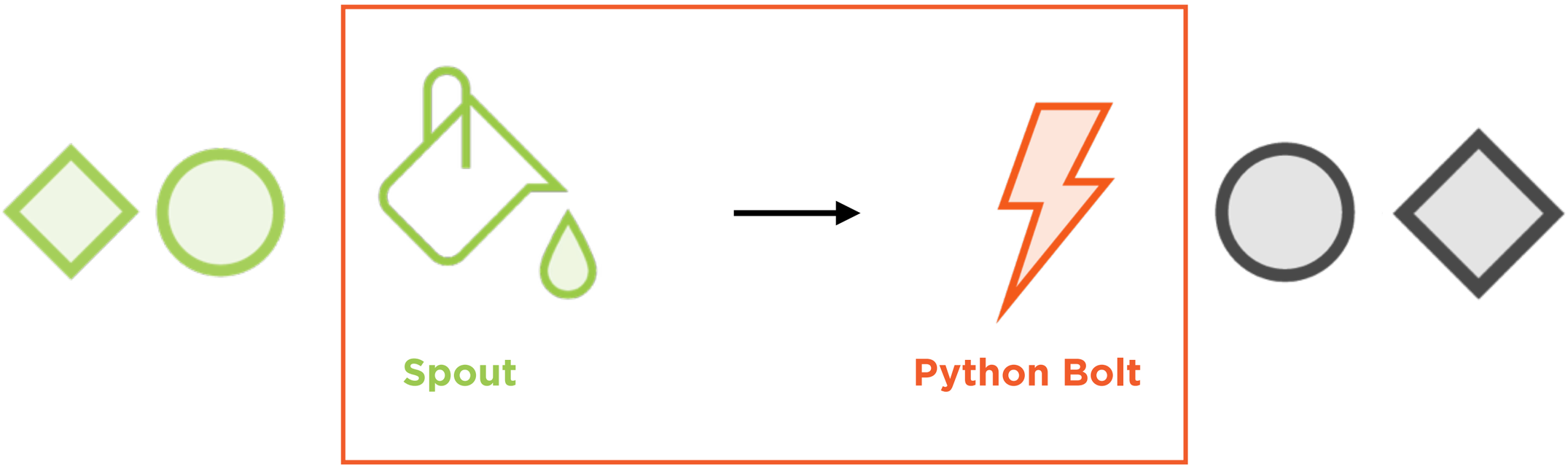
Use a real-time Storm application

Integrating Storm with Python

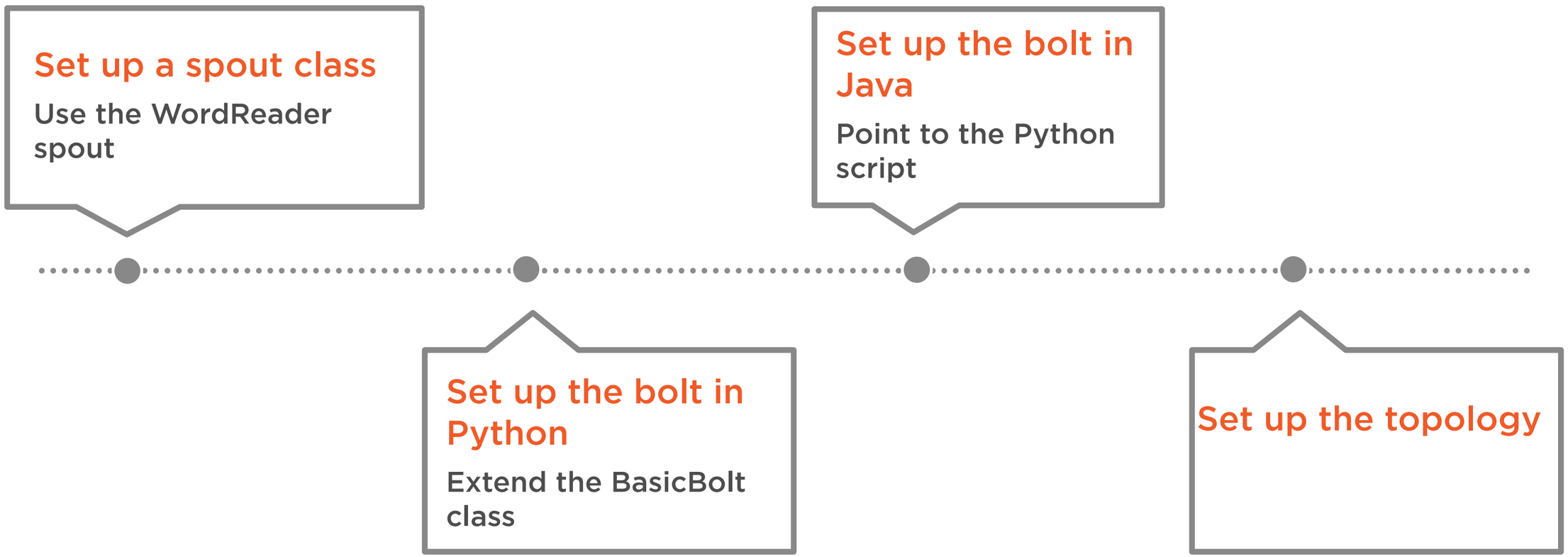
Storm Multilang Protocol



Storm Topology



Integrating with Python



Demo

Set up a topology which runs with Python

Predicting Sentiment in Real-time

Sentiment Analysis

Product Launches

Political Candidates

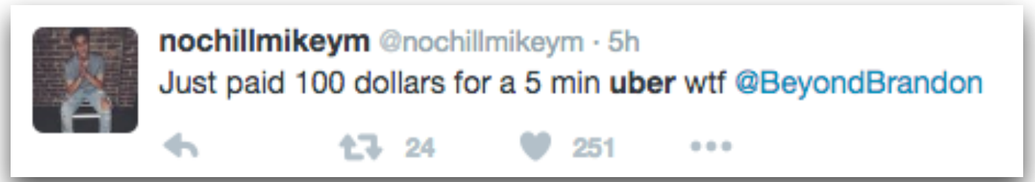
Financial Reports

Sentiment Analysis

Positive

This comment is positive

Negative



These comments are negative

Sentiment Analysis

Positive or **Negative**

Identify the polarity of a comment

Machine Learning Process

Training Phase

Train a model using historical data

Test Phase

Apply the model on new data

Training Phase

Data Representation

Text 1	Positive
Text 2	Positive
Text 3	Negative
Text 4	Positive
Text 5	Negative
Text 6	Positive
.	.
.	.
.	.
.	.
Text 100	Positive

Training Data

Training Phase

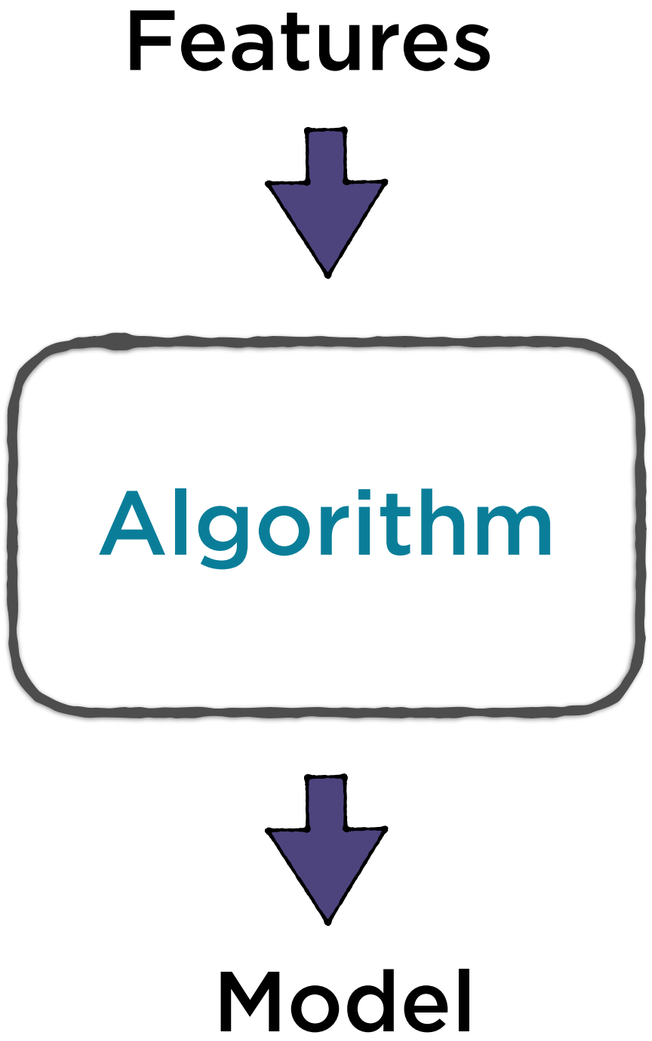
Data Representation

(1,0,1,1..0)	Positive
(1,1,0,1..0)	Positive
(0,0,1,0..0)	Negative
(0,0,0,1..0)	Positive
(1,1,1,1..0)	Negative
(0,0,1,1..1)	Positive
. . . .	. . . .
(1,0,1,1..0)	Positive

Extracting Features

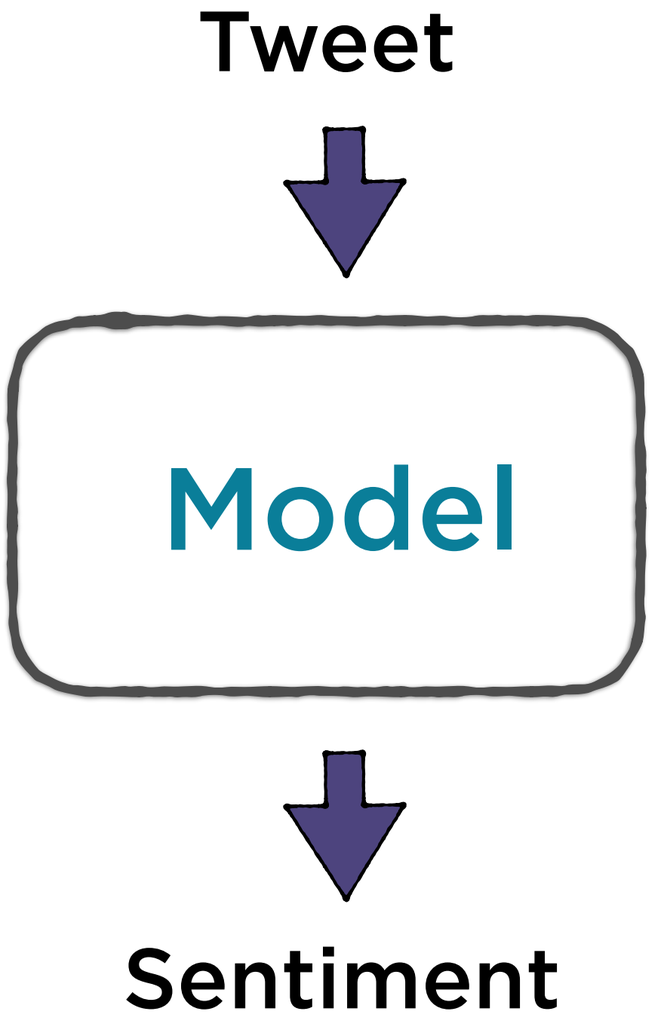
Training Phase

Training a Model



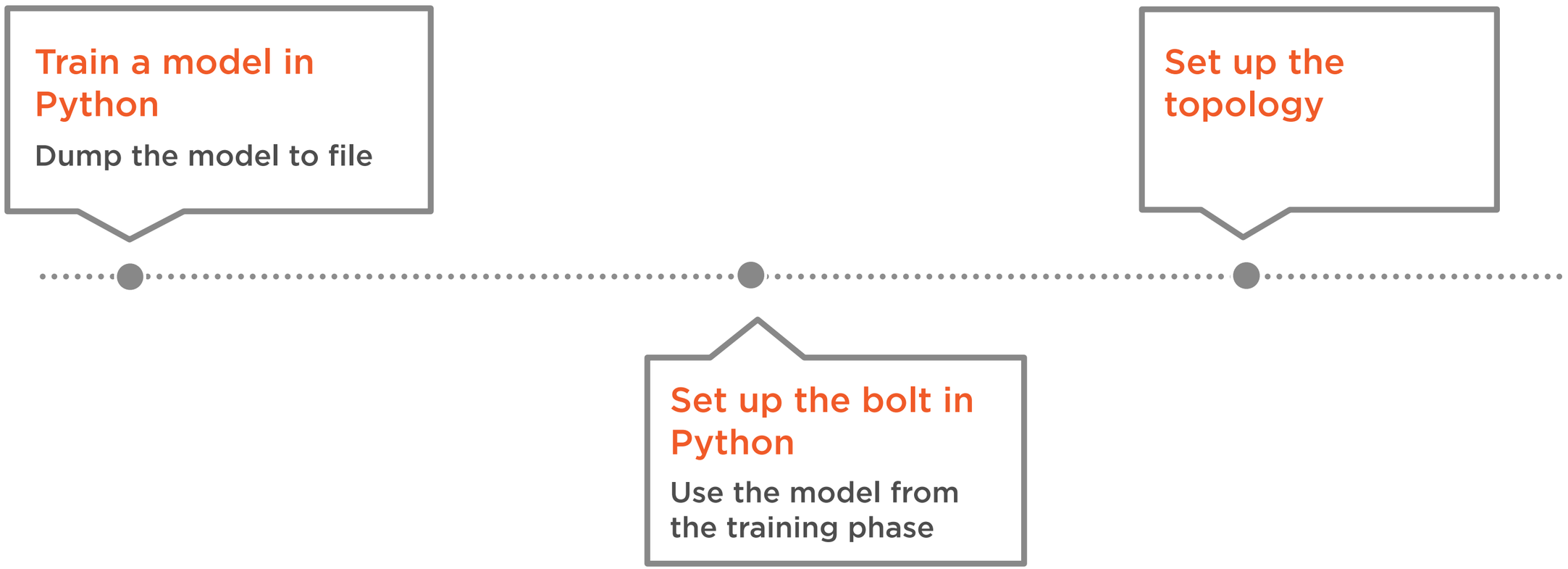
Test Phase

Predict Sentiment in Real-time



Sentiment Analysis with Storm

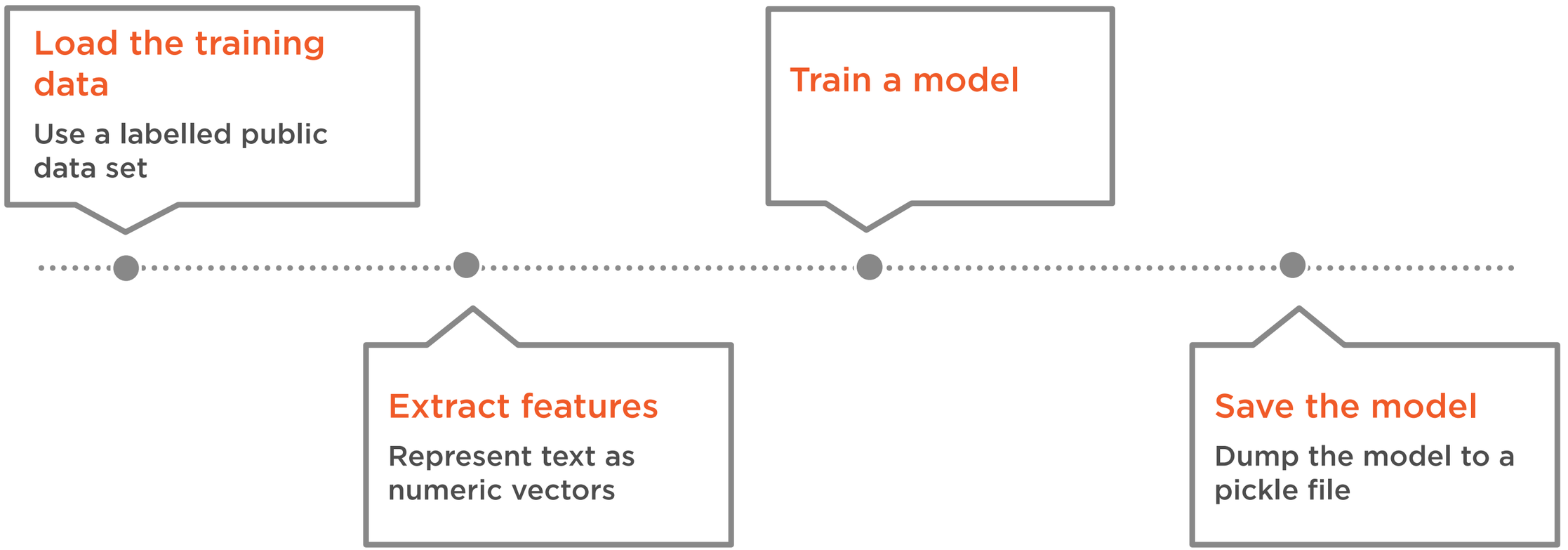
Sentiment Analysis with Storm



Demo

Training a model in Python

Training a Model



Training a Model

Load the training data

Use a labelled public data set

```
lines=pd.read_csv(dataFile,sep="\t",header=None,
                  names=['sentence','label'])
```

Load a Pandas DataFrame

```
lines=pd.read_csv(dataFile,sep="\t",header=None,
                  names=['sentence','label'])
```

Load a Pandas DataFrame

```
lines=pd.read_csv(dataFile,sep="\t",header=None,
                  names=['sentence','label'])
```

Load a Pandas DataFrame

Training a Model

Load the training data

Use a labelled public data set

Extract features

Represent text as numeric vectors

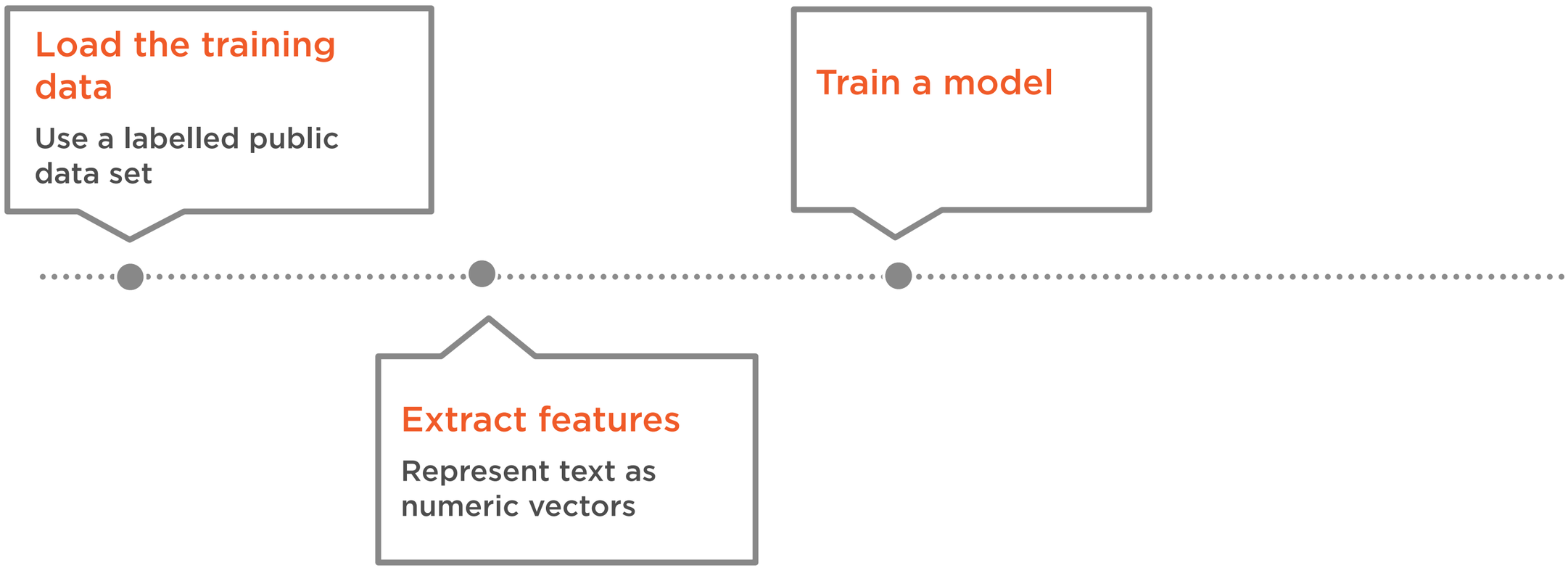
```
count_vectorizer = CountVectorizer(binary='true')
train_documents = count_vectorizer.fit_transform(lines['sentence'])
```

Convert Text to Numeric Vectors

```
count_vectorizer = CountVectorizer(binary='true')
train_documents = count_vectorizer.fit_transform(lines['sentence'])
```

Convert Text to Numeric Vectors

Training a Model



```
classifier = BernoulliNB().fit(train_documents,lines['label'])
```

Train the Model

```
classifier = BernoulliNB().fit(train_documents, lines['label'])
```

Train the Model

```
classifier = BernoulliNB().fit(train_documents, lines['label'])
```

Train the Model

```
classifier.predict(count_vectorizer.transform(["this is the worst movie"]))
```

Test the Model

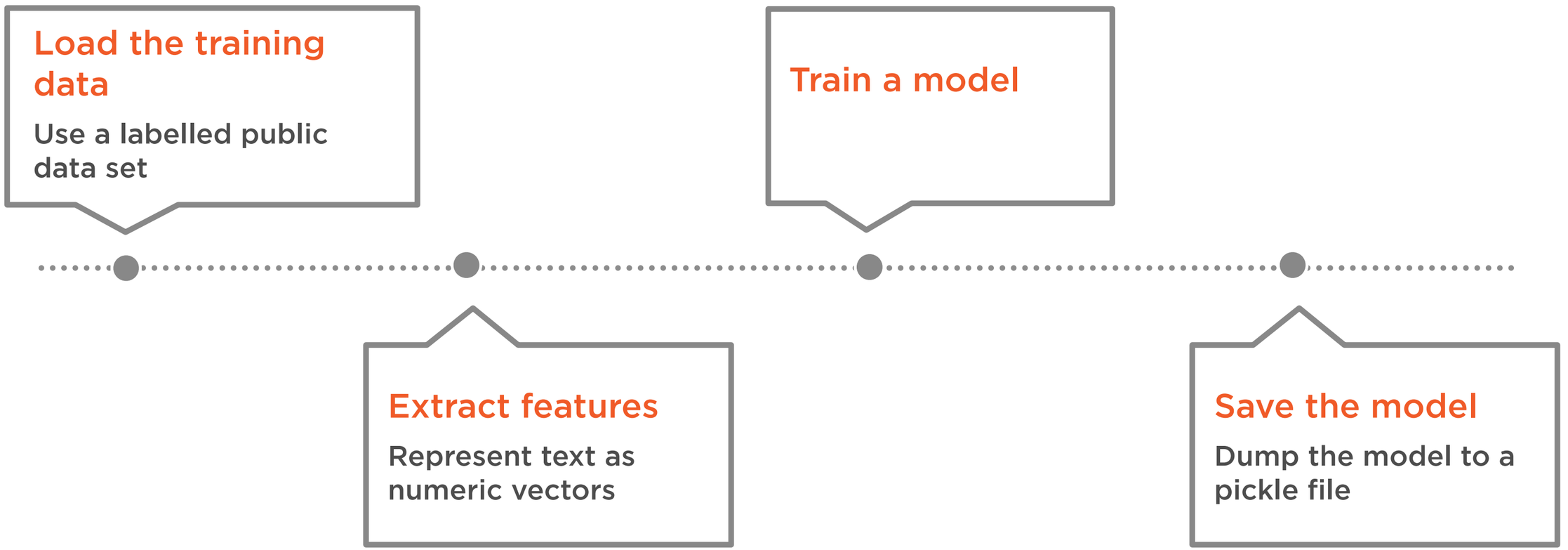
```
classifier.predict(count_vectorizer.transform(["this is the worst movie"]))
```

Test the Model

```
classifier.predict(count_vectorizer.transform(["this is the worst movie"])
```

Test the Model

Training a Model



```
joblib.dump(classifier, 'SAModel.pkl')
joblib.dump(count_vectorizer,'Vectorizer.pkl')
```

Save the Model

```
joblib.dump(classifier, 'SAModel.pkl')
joblib.dump(count_vectorizer,'Vectorizer.pkl')
```

Save the Model

```
joblib.dump(classifier, 'SAModel.pkl')
joblib.dump(count_vectorizer,'Vectorizer.pkl')
```

Save the Model

Demo

Setting up a topology to predict the sentiment of a sentence

Summary

Process data streams using the Trident API

Use Trident for Distributed Remote Procedure Calls i.e. DRPC

Maintain the state of a data stream

Query a state object