Realtime predictive analytics

using scikit-learn & RabbitMQ

Michael Becker

#pycon

@beckerfuffle

Who Is This Guy?

Data guy @ AWeber

@beckerfuffle

beckerfuffle.com

These slides and more @ github.com/mdbecker

What My Coworkers Think I Do

$$h_{w,b}(x) = g(w^T x + b)$$

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b)$$

$$\hat{\gamma} = \min_{i=1,\ldots,m} \hat{\gamma}^{(i)}$$

$$w^T \left(x^{(i)} - \gamma^{(i)} \frac{w}{||w||} \right) + b = 0$$

What I Actually Do

```
from sklearn.svm import SVC
```

What I'll Cover

- Scikit-learn overview

What I'll Cover

- Scikit-learn overview
- Model Distribution

What I'll Cover

- Scikit-learn overview
- Model Distribution
- Data flow

What I'll Cover

- Scikit-learn overview
- Model Distribution
- Data flow
- RabbitMQ

What I'll Cover

- Scikit-learn overview
- Model Distribution
- Data flow
- RabbitMQ
- Demo

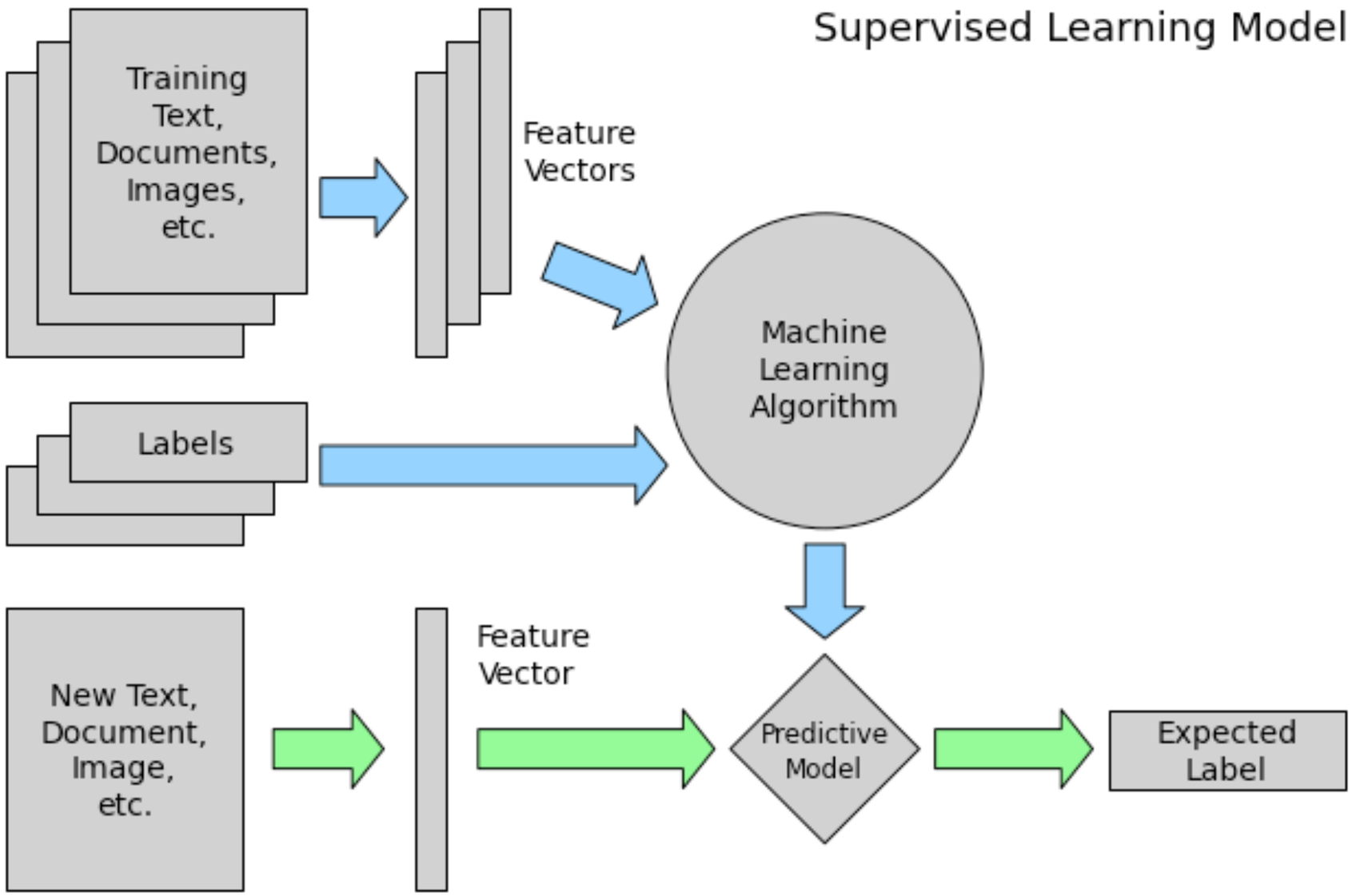
What I'll Cover

- Scikit-learn overview
- Model Distribution
- Data flow
- RabbitMQ
- Demo
- Scalability

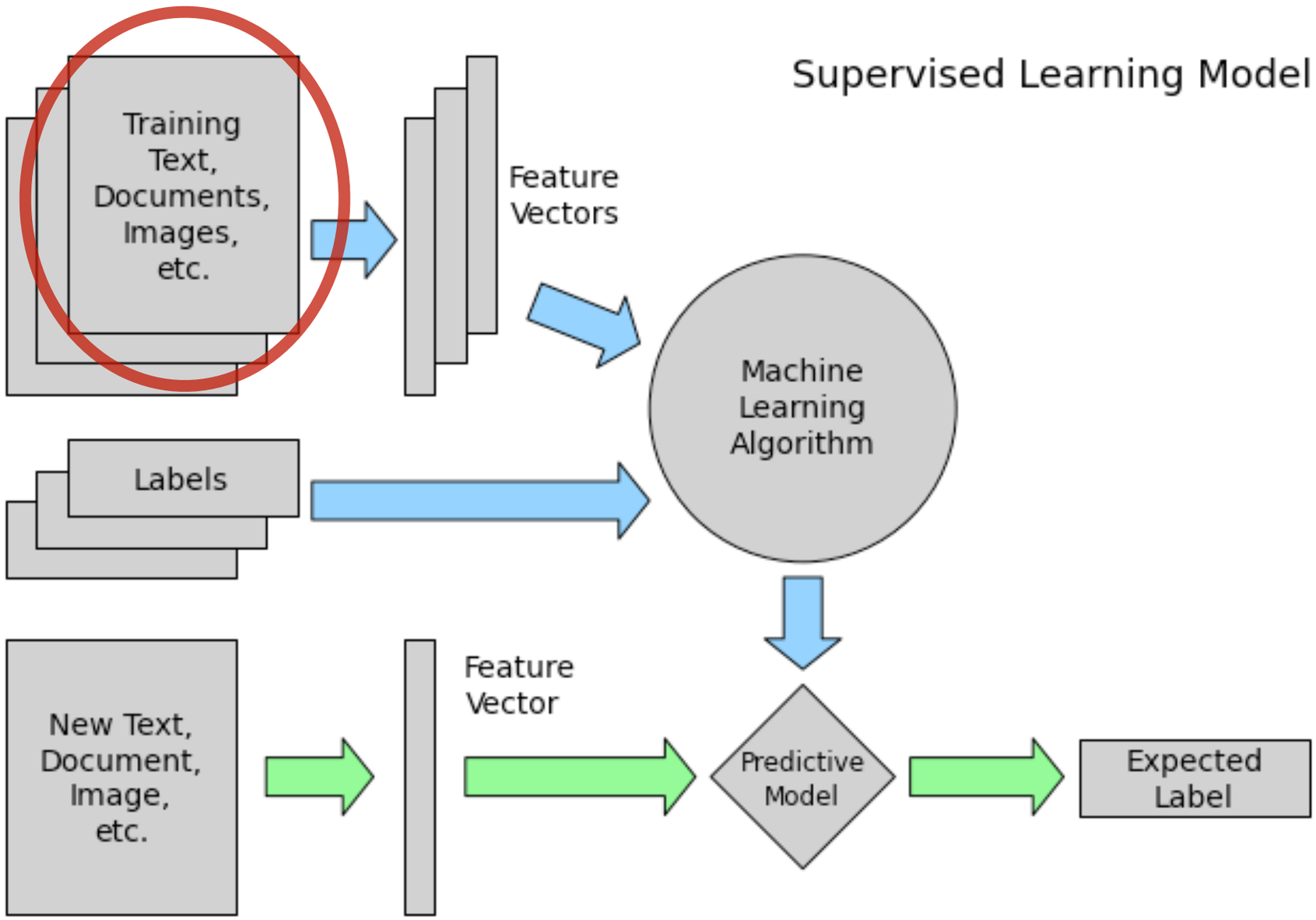
What I'll Cover

- Scikit-learn overview
- Model Distribution
- Data flow
- RabbitMQ
- Demo
- Scalability
- Other considerations

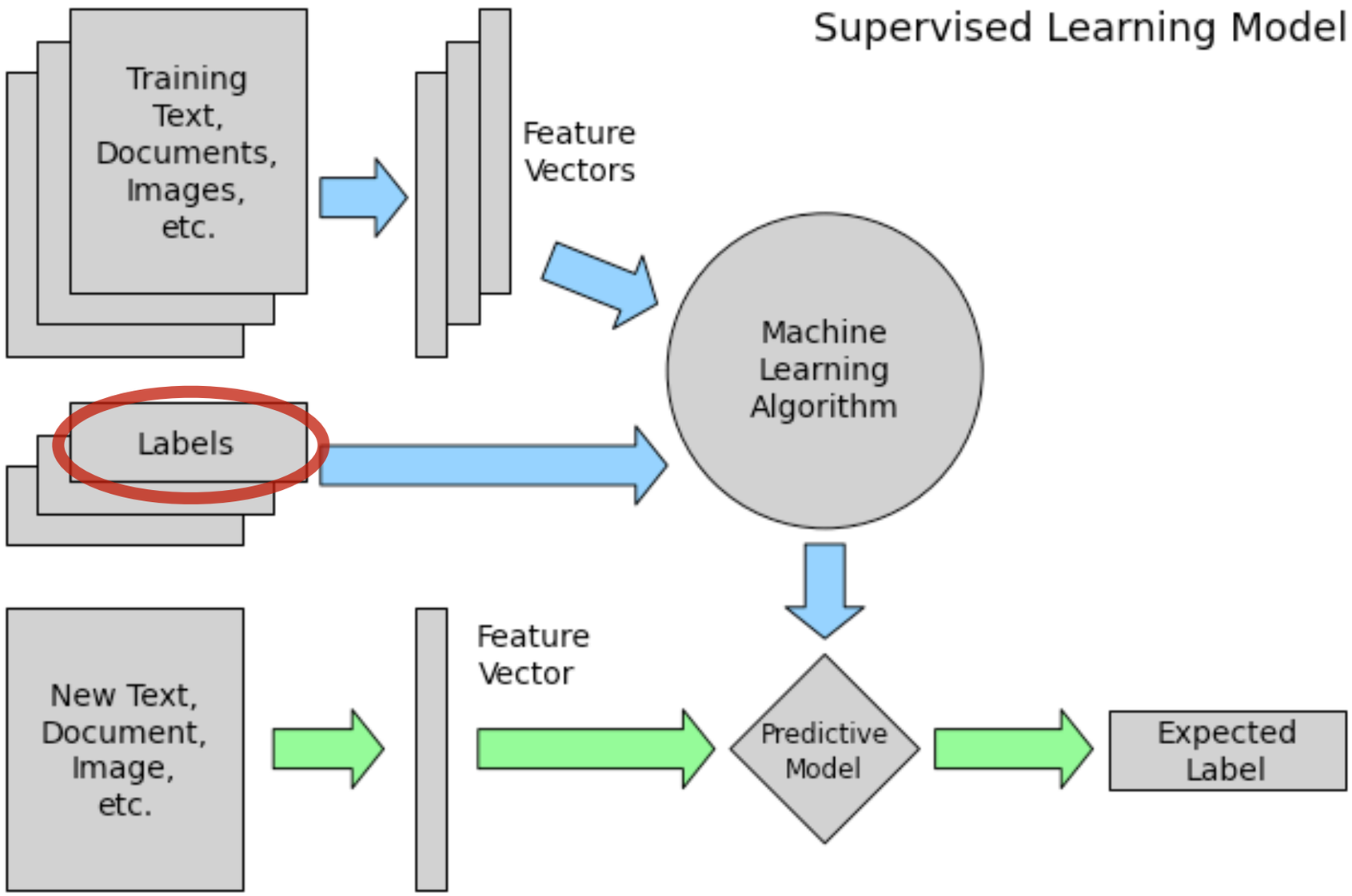
Supervised Learning



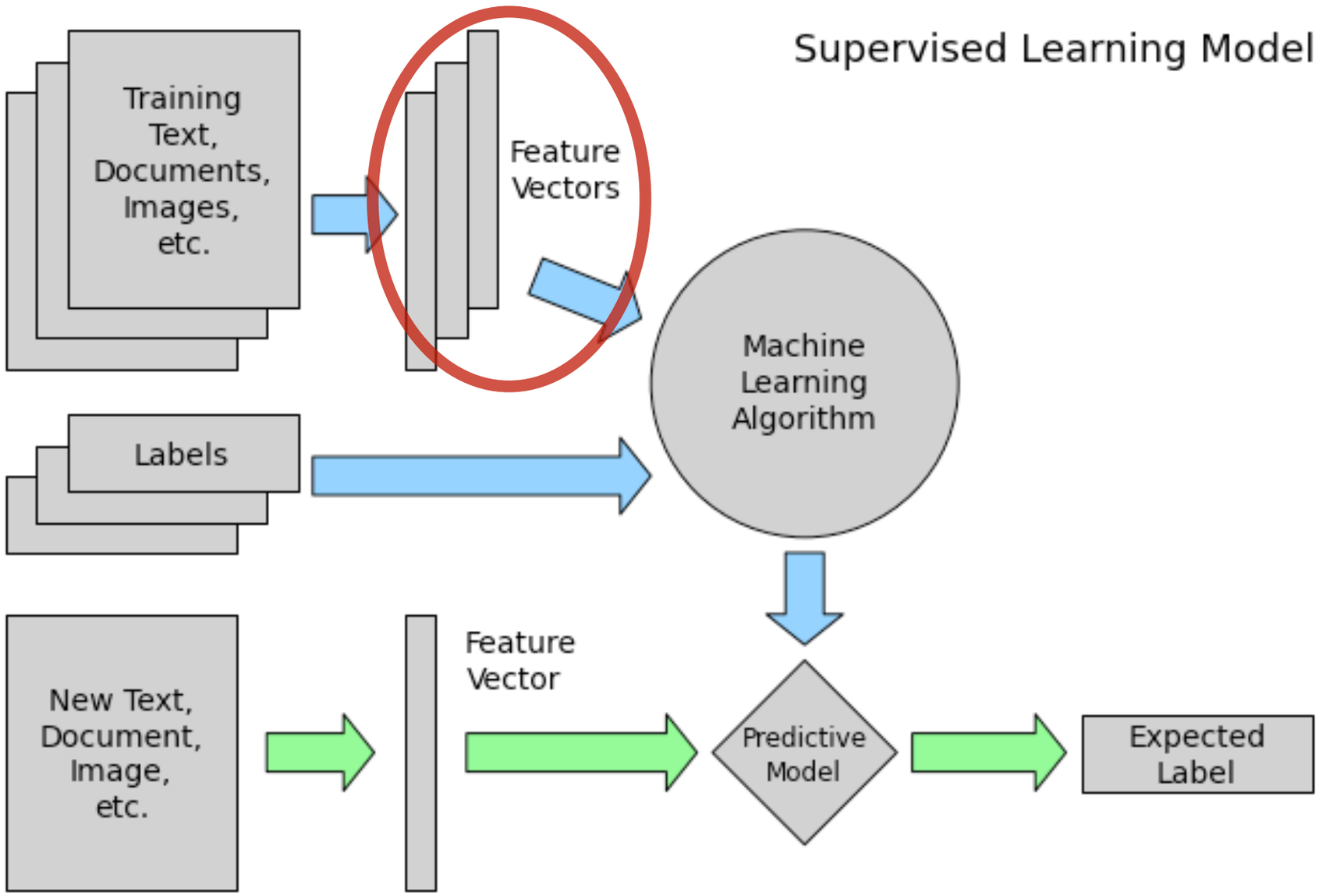
Supervised Learning



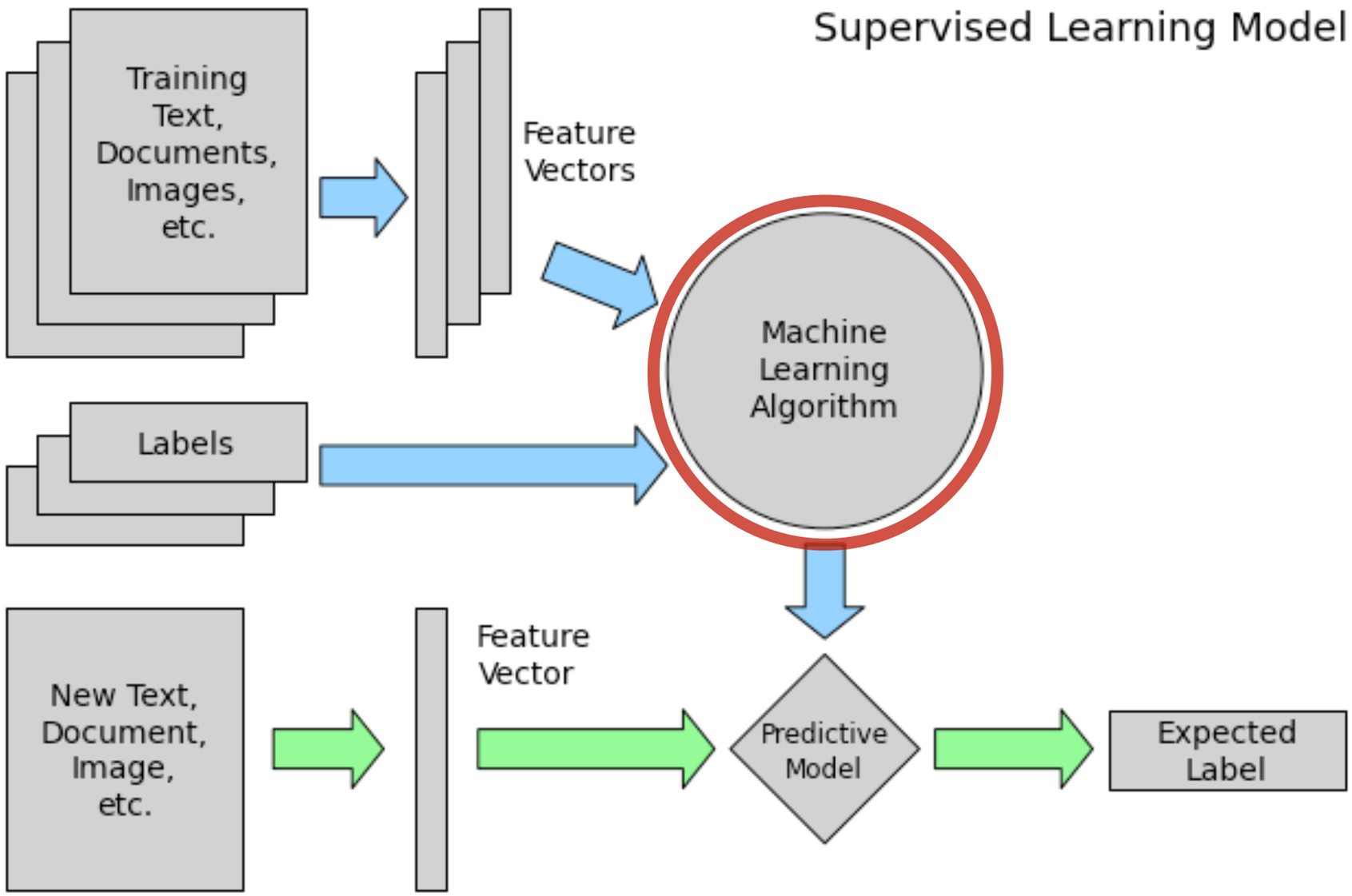
Supervised Learning



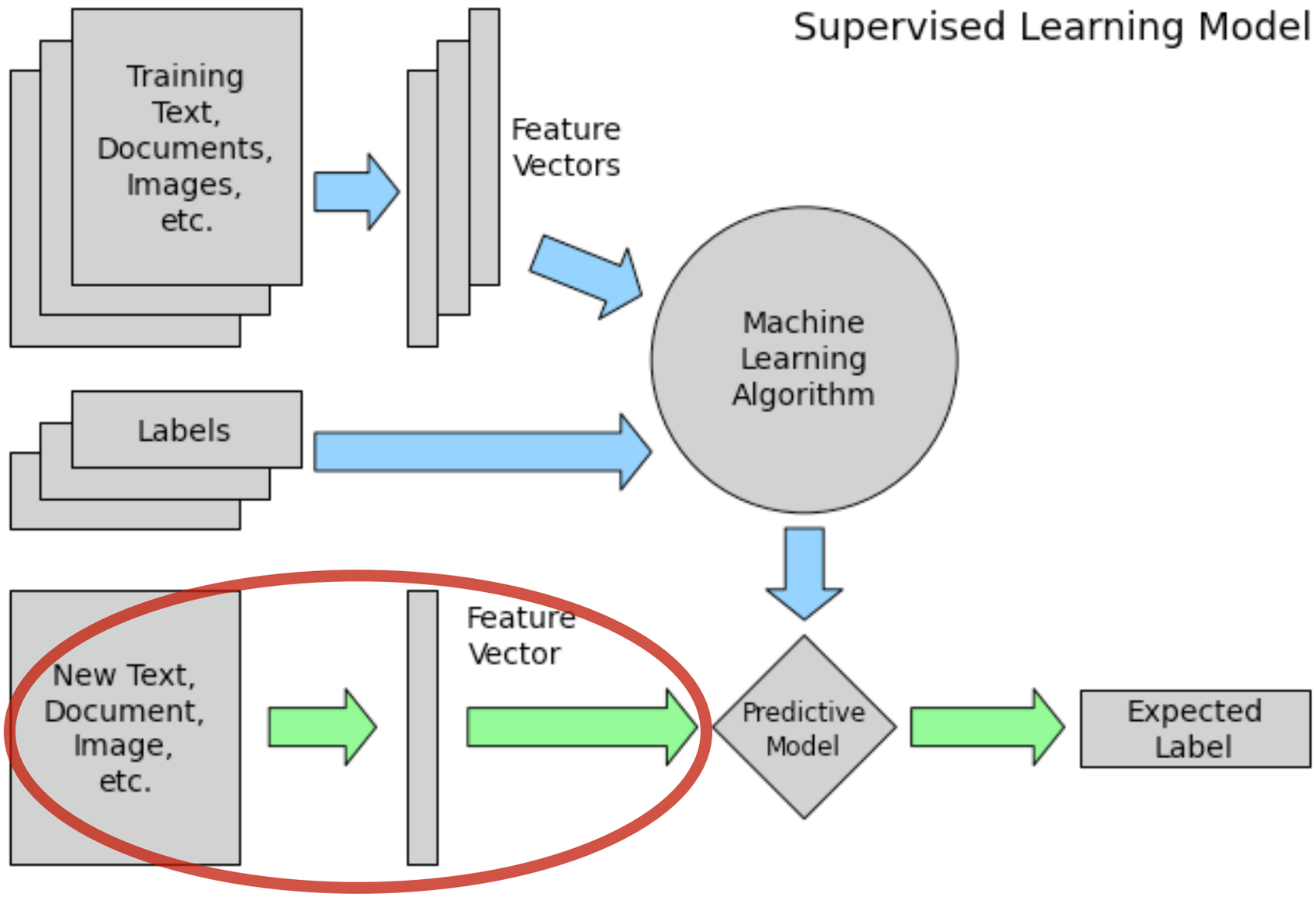
Supervised Learning



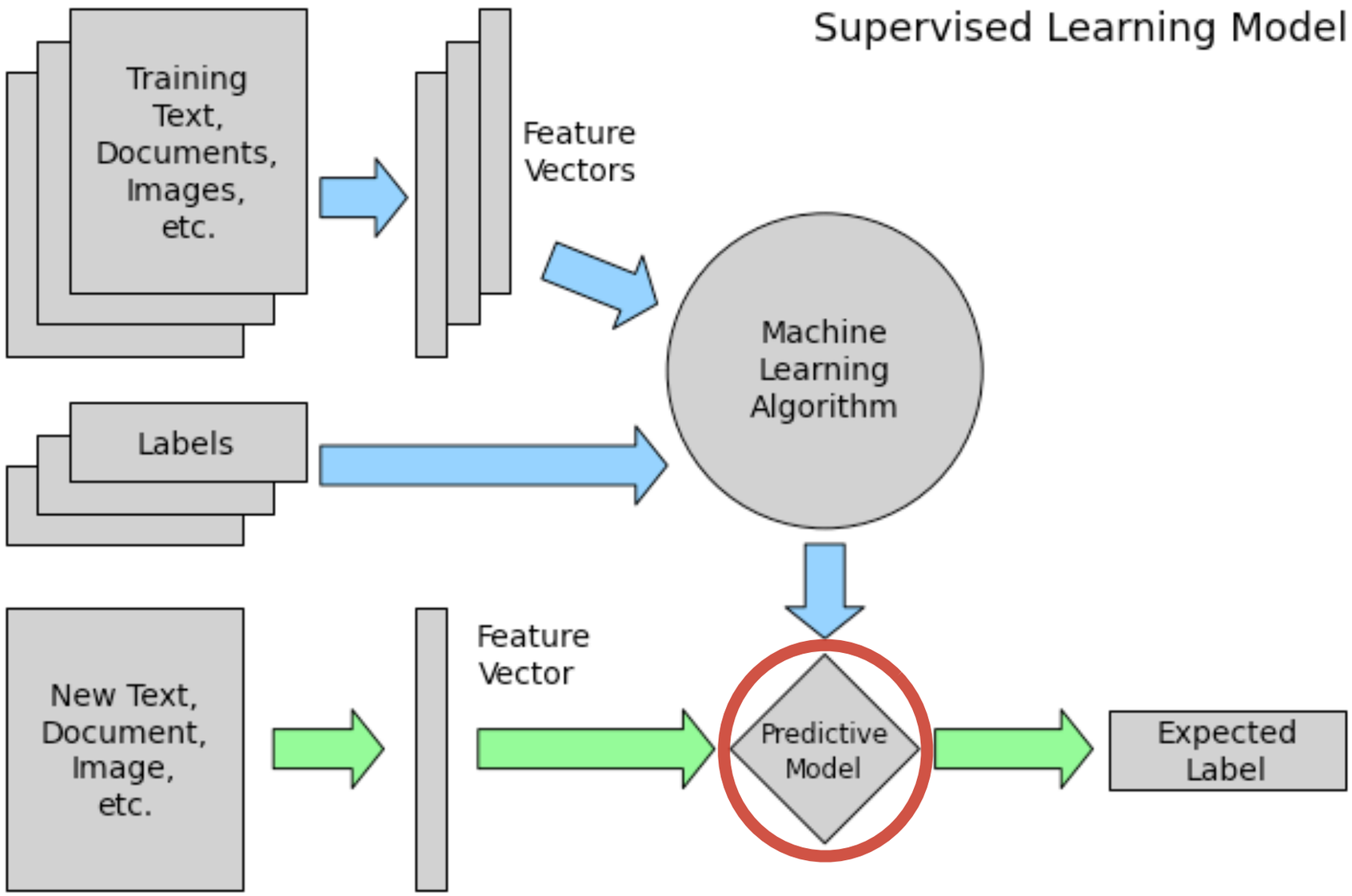
Supervised Learning



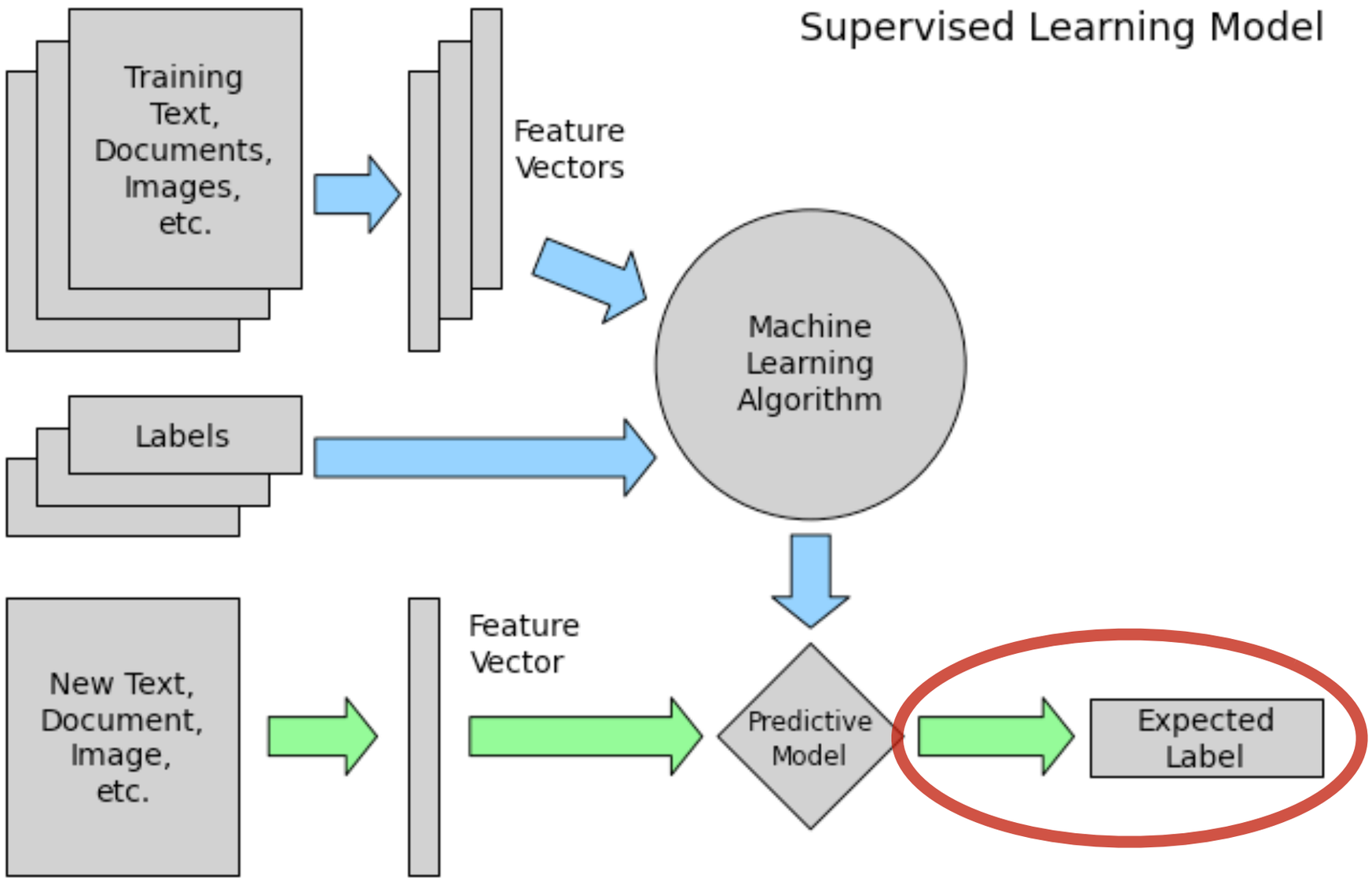
Supervised Learning



Supervised Learning



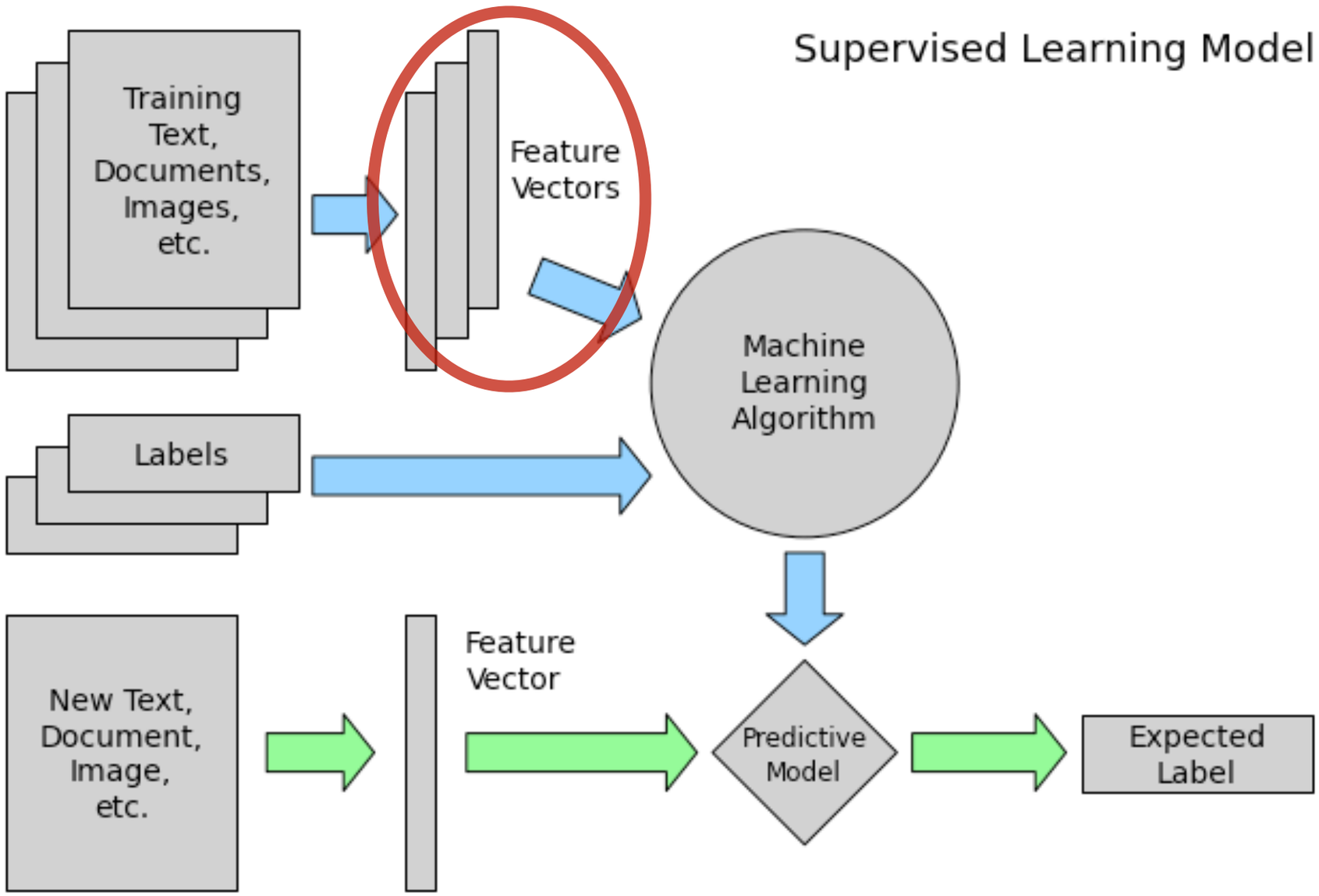
Supervised Learning



38 Top Wikipedias

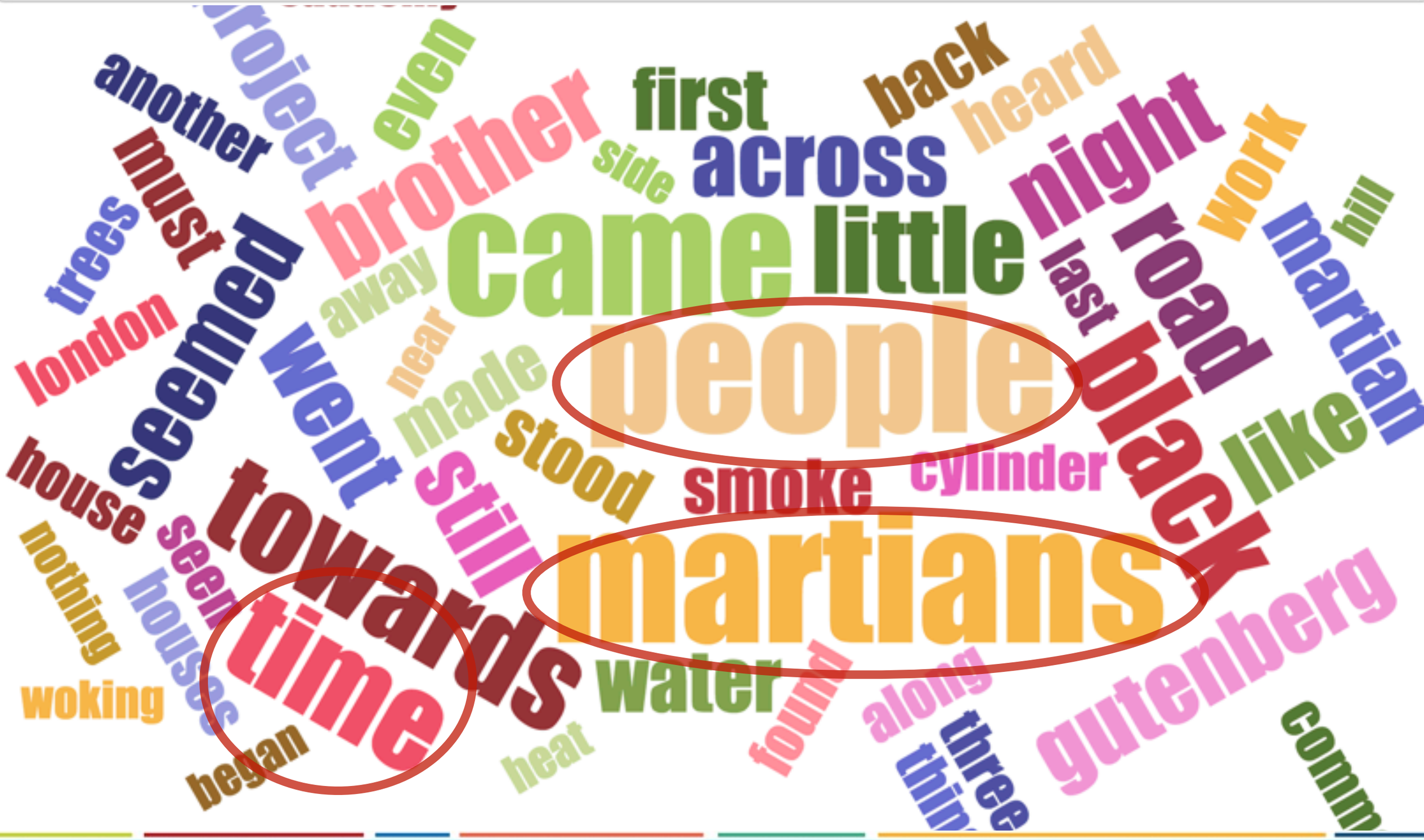
Arabic: العربية
Bulgarian: Български
Catalan: Català
Czech: Čeština
Danish: Dansk
German: Deutsch
English: English
Spanish: Español
Estonian: Eesti
Basque: Euskara
Persian: فارسی
Finnish: Suomi
French: Français
Hebrew: עברית
Hindi: हिन्दी
Croatian: Hrvatski
Hungarian: Magyar
Indonesian: Bahasa Indonesia
Italian: Italiano
Japanese: 日本語
Kazakh: Қазақша
Korean: 한국어
Lithuanian: Lietuvių
Malay Bahasa: Melayu
Dutch: Nederlands
Norwegian: Norsk (Bokmål)
Polish: Polski
Portuguese: Português
Romanian: Română
Russian: Русский
Slovak: Slovenčina
Slovenian: Slovenščina
Serbian: Српски / Srpski
Swedish: Svenska
Turkish: Türkçe
Ukrainian: Українська
Vietnamese: Tiếng Việt
Waray-Waray: Winaray

Vectorizing Text



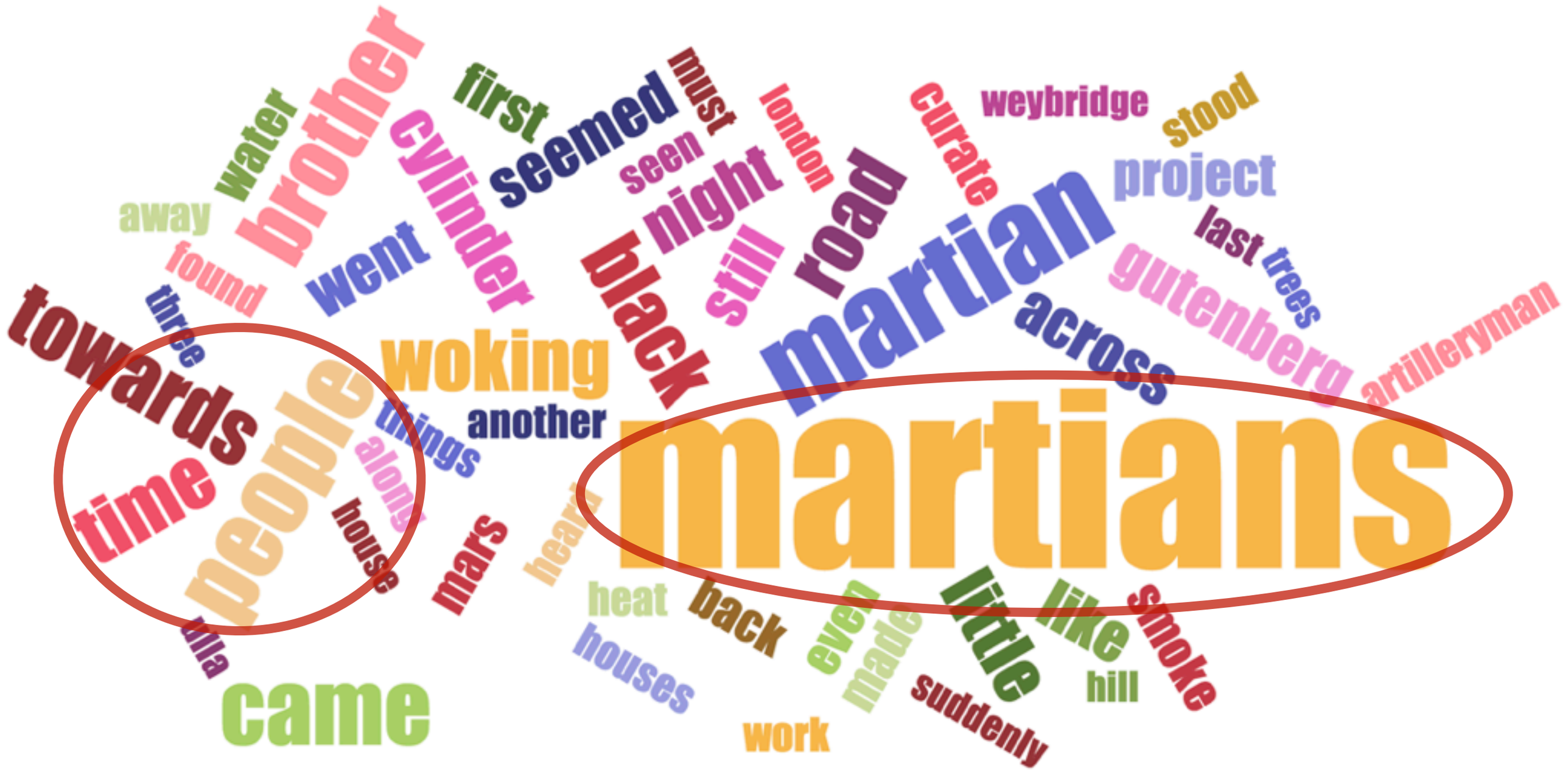

suddenly
project
even
another
brother
first
back
heard
side
across
night
must
trees
came
little
work
road
martian
hill
away
last
london
people
near
made
stood
cylinder
like
house
smoke
towards
still
martians
seen
time
water
gutenberg
woking
houses
began
found
along
heat

```
CountVectorizer(analyzer='word', ngram_range=(1, 1))
```




martians
martian
brother
cylinder
seemed
first
must
water
away
found
went
black
night
still
road
curate
weybridge
stood
project
last
trees
gutenberg
artilleryman
across
three
towards
people
woking
another
things
along
house
mars
heard
time
ulla
came
heat
back
houses
even
made
little
like
smoke
hill
suddenly
work
seen
london

```
TfidfVectorizer(analyzer='word', ngram_range=(1, 1))
```



The Model

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
```

```
vect = TfidfVectorizer(analyzer='char', ngram_range=(2, 3))
clf = SVC()
text_clf = Pipeline([
    ('vect', vect),
    ('clf', clf),
])
text_clf = text_clf.fit_transform(X_train, y_train)
```

X_train

```
for x in X_train[:10]:
    print ' '.join(unicode(x, 'utf8')[:70].split())
```

```
Szöul Szöul (서울 특별시 Sŏul T'ŭkpyŏlsi, szoros átírásban: Szoul thukpjol
Sean Connery Cecil B. DeMille Award (1996) | baftaawards Beste hov
Nemecká demokratická republika Nemecká demokratická republika (NDR, h
Plastic A plastic material is any of a wide range of synthetic or sem
Phaistose ketas Phaistose ketas on põletatud savist ketas, mille leid
Soustava SI Soustava SI (zkratka z francouzského Le Système Internati
Баку Баку () је главни град Азербејџана. Налази се у јужном делу полу
Gottlob Frege nume Friedrich Ludwig Gottlob Frege
Belgrado Belgrado (Београд / Beograd em servo-croata ouça ) é a capi
Cálculo infinitesimal El cálculo infinitesimal o cálculo de infinites
```

y_train

```
y
```

```
['uk', 'cs', 'ko', 'es', 'war', 'de', 'de', 'pl', 'ar', 'cs']
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train = le.fit_transform(y)
y_train
```

```
array([35,  3, 21, ...,  8, 18, 30])
```

Dimensionality Reduction

```
from sklearn.decomposition import RandomizedPCA
```

```
vect = TfidfVectorizer(analyzer='char', ngram_range=(2, 3)
pca = RandomizedPCA(n_components=50, whiten=True)
clf = SVC()
text_clf = Pipeline([
    ('vect', vect),
    ('pca', pca),
    ('clf', clf),
])
text_clf = text_clf.fit_transform(X_train, y_train)
```

Dimensionality Reduction

```
vect = TfidfVectorizer(analyzer='char', ngram_range=(2, 3)
X1_train = vect.fit_transform(X_train)
X1_train.shape
```

```
(47221, 1048576)
```

Dimensionality Reduction

```
vect = TfidfVectorizer(analyzer='char', ngram_range=(2, 3)
X1_train = vect.fit_transform(X_train)
X1_train.shape
```

```
(47221, 1048576)
```

```
pca = RandomizedPCA(n_components=50, whiten=True)
X2_train = pca.fit_transform(X1_train)
X2_train.shape
```

```
(47221, 50)
```

Dimensionality Reduction

```
pca.explained_variance_ratio_
```

```
array([ 0.36366957,  0.0984124 ,  0.04520123,  0.04459129,  0.03954548,
        0.03202782,  0.02943691,  0.02429984,  0.01907511,  0.01890865,
        0.01858178,  0.0171858 ,  0.016482  ,  0.01596849,  0.01584665,
        0.01499711,  0.01378617,  0.01368058,  0.01190577,  0.01173788,
        0.01131731,  0.01045798,  0.01029921,  0.01004662,  0.00951628,
        0.00803825,  0.00728402,  0.00700877,  0.00667922,  0.00664124,
        0.00574287,  0.00548161,  0.00500291,  0.00433605,  0.003948  ,
        0.00283822,  0.00241628,  0.00189858,  0.00168776,  0.00160769,
        0.001431  ,  0.00137901,  0.00132682,  0.00129177,  0.00126926,
        0.0012235 ,  0.00117239,  0.00114645,  0.00110811,  0.00106229])
```

Dimensionality Reduction

```
pca.explained_variance_ratio_
```

```
array([ 0.36366957,  0.0984124 ,  0.04520123,  0.04459129,  0.03954548,
        0.03202782,  0.02943691,  0.02429984,  0.01907511,  0.01890865,
        0.01858178,  0.0171858 ,  0.016482  ,  0.01596849,  0.01584665,
        0.01499711,  0.01378617,  0.01368058,  0.01190577,  0.01173788,
        0.01131731,  0.01045798,  0.01029921,  0.01004662,  0.00951628,
        0.00803825,  0.00728402,  0.00700877,  0.00667922,  0.00664124,
        0.00574287,  0.00548161,  0.00500291,  0.00433605,  0.003948  ,
        0.00283822,  0.00241628,  0.00189858,  0.00168776,  0.00160769,
        0.001431  ,  0.00137901,  0.00132682,  0.00129177,  0.00126926,
        0.0012235 ,  0.00117239,  0.00114645,  0.00110811,  0.00106229])
```

```
pca.explained_variance_ratio_.sum()
```

```
0.999999999999999
```

Don’t Do What I Just Did!

sklearn.decomposition.TruncatedSVD

class `sklearn.decomposition.`**`TruncatedSVD`**(*n_components=2, algorithm='randomized', n_iterations=5, random_state=None, tol=0.0*)

Dimensionality reduction using truncated SVD (aka LSA).

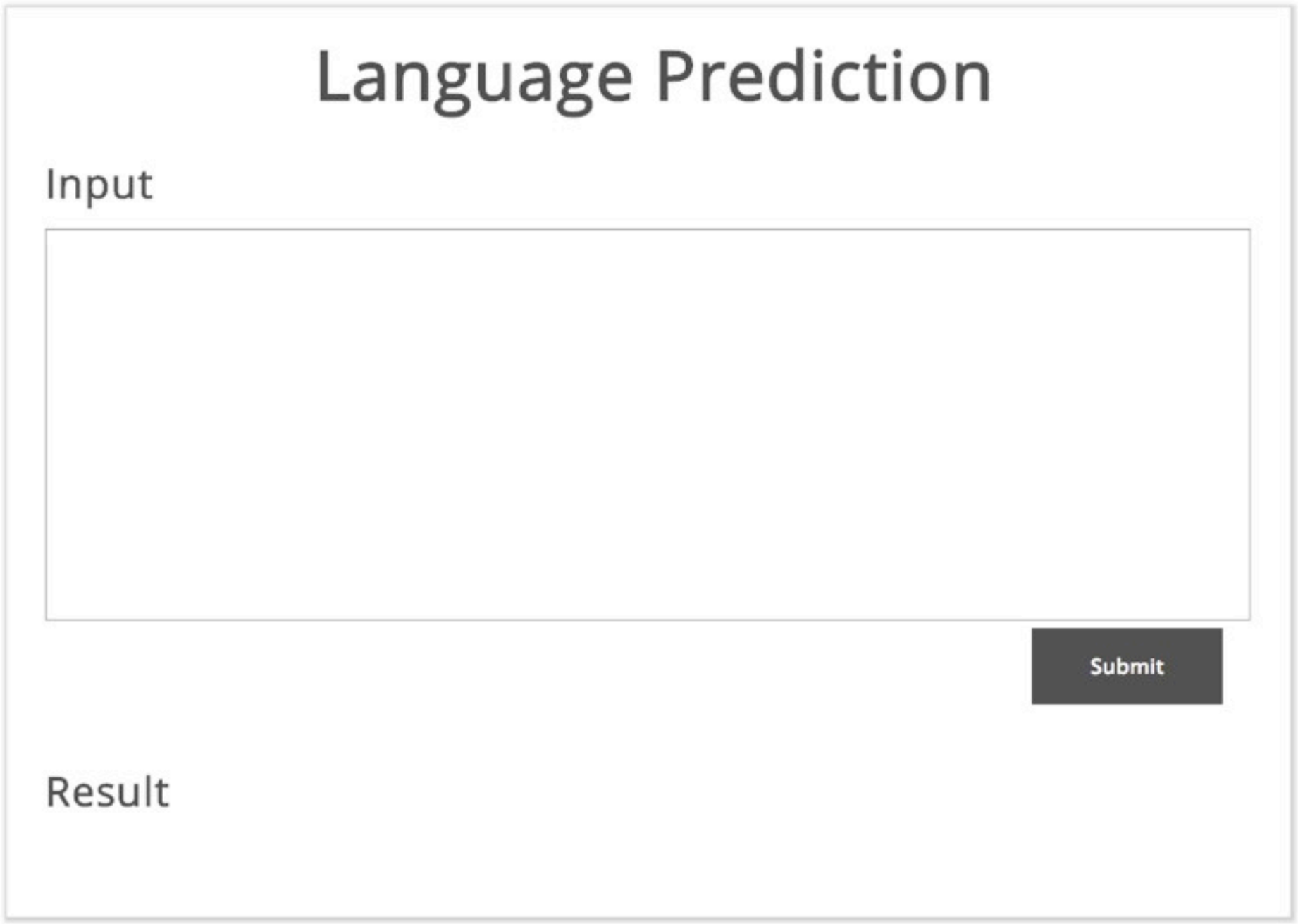
This transformer performs linear dimensionality reduction by means of truncated singular value decomposition (SVD). It is very similar to PCA, but operates on sample vectors directly, instead of on a covariance matrix. This means it can work with scipy.sparse matrices efficiently.

In particular, truncated SVD works on term count/tf–idf matrices as returned by the vectorizers in sklearn.feature_extraction.text. In that context, it is known as latent semantic analysis (LSA).

Distributing The Model

Data Input

The Client



Message Loss

Enter RabbitMQ

Flexible Routing

Reliability

Clustering

HA Queues

Many clients

Enter RabbitMQ

Flexible Routing

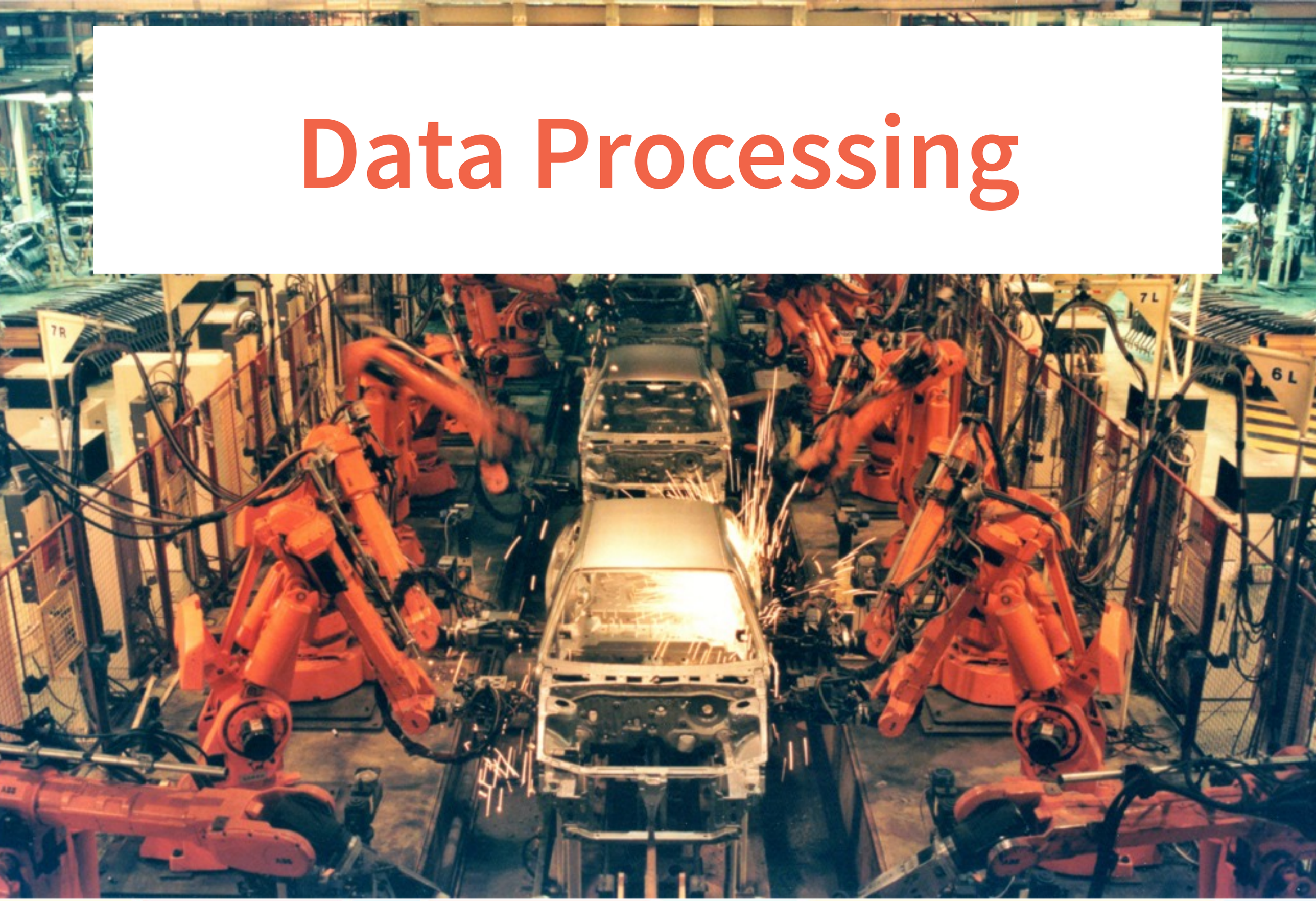
Reliability

Clustering

HA Queues

Many clients

Data Processing



The Consumer

```
class LanguagePredictorWorker(object):

    classifier, label_encoder = load_pickled_files(clf)
    def __init__(self):
        subscribe_to_queue()
        self.language_coll = get_db_collection()

    def process_event(self, body, message):
        text = body['text']
        _id = body['id']
        language = self.predict_language_for_text(text)
        result = self.language_coll.update(
            {'_id': ObjectId(_id), 'text_input': text},
            {'$set': {'language': language}},
        )
        message.ack()

    def predict_language_for_text(self, text):
        lang_vector = self.classifier.predict([text])
        lang_labels = self.label_encoder.inverse_transform(lang_vector)
        return lang_labels[0]

worker = LanguagePredictorWorker()
worker.main()
```

The Consumer

```
class LanguagePredictorWorker(object):

    classifier, label_encoder = load_pickled_files(clf)
    def __init__(self):
        subscribe_to_queue()
        self.language_coll = get_db_collection()

    def process_event(self, body, message):
        text = body['text']
        _id = body['id']
        language = self.predict_language_for_text(text)
        result = self.language_coll.update(
            {'_id': ObjectId(_id), 'text_input': text},
            {'$set': {'language': language}},
        )
        message.ack()

    def predict_language_for_text(self, text):
        lang_vector = self.classifier.predict([text])
        lang_labels = self.label_encoder.inverse_transform(lang_vector)
        return lang_labels[0]

worker = LanguagePredictorWorker()
worker.main()
```

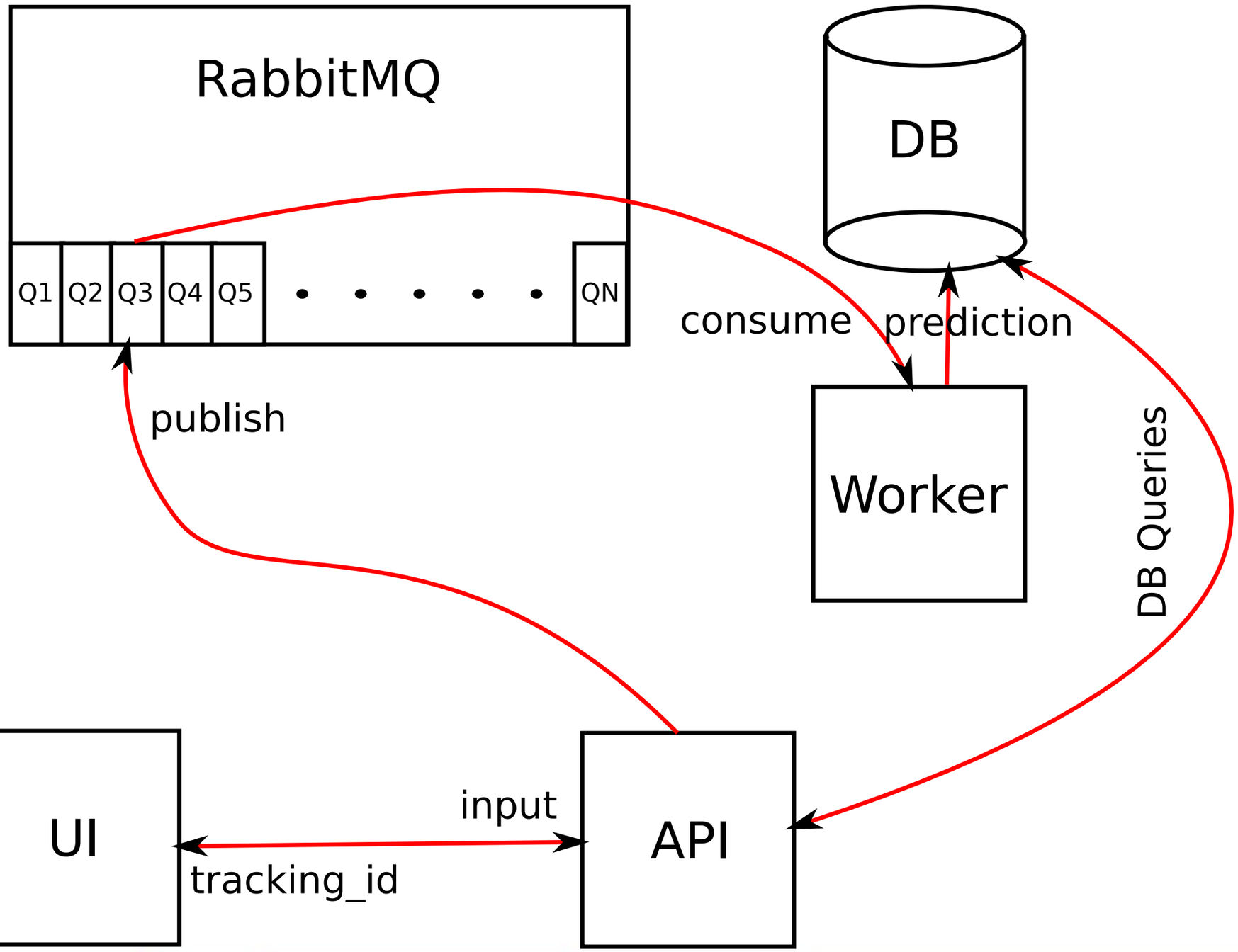
The Consumer

```
class LanguagePredictorWorker(object):

    classifier, label_encoder = load_pickled_files(clf)
    def __init__(self):
        subscribe_to_queue()
        self.language_coll = get_db_collection()

    def process_event(self, body, message):
        text = body['text']
        _id = body['id']
        language = self.predict_language_for_text(text)
        result = self.language_coll.update(
            {'_id': ObjectId(_id), 'text_input': text},
            {'$set': {'language': language}},
        )
        message.ack()

    def predict_language_for_text(self, text):
        lang_vector = self.classifier.predict([text])
        lang_labels = self.label_encoder.inverse_transform(lang_vector)
        return lang_labels[0]

worker = LanguagePredictorWorker()
worker.main()
```

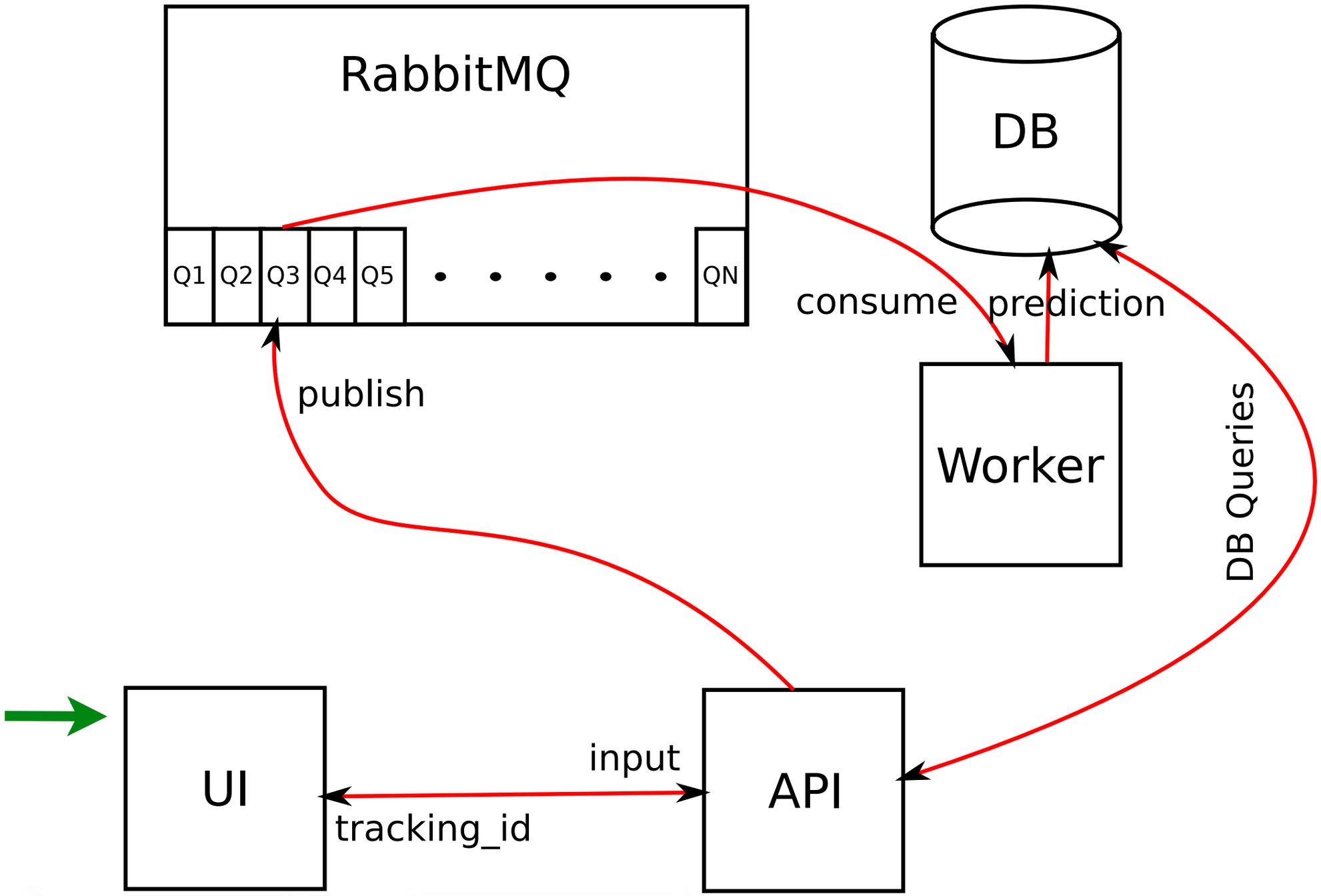
The Consumer

```
class LanguagePredictorWorker(object):

    classifier, label_encoder = load_pickled_files(clf)
    def __init__(self):
        subscribe_to_queue()
        self.language_coll = get_db_collection()

    def process_event(self, body, message):
        text = body['text']
        _id = body['id']
        language = self.predict_language_for_text(text)
        result = self.language_coll.update(
            {'_id': ObjectId(_id), 'text_input': text},
            {'$set': {'language': language}},
        )
        message.ack()

    def predict_language_for_text(self, text):
        lang_vector = self.classifier.predict([text])
        lang_labels = self.label_encoder.inverse_transform(lang_vector)
        return lang_labels[0]

worker = LanguagePredictorWorker()
worker.main()
```

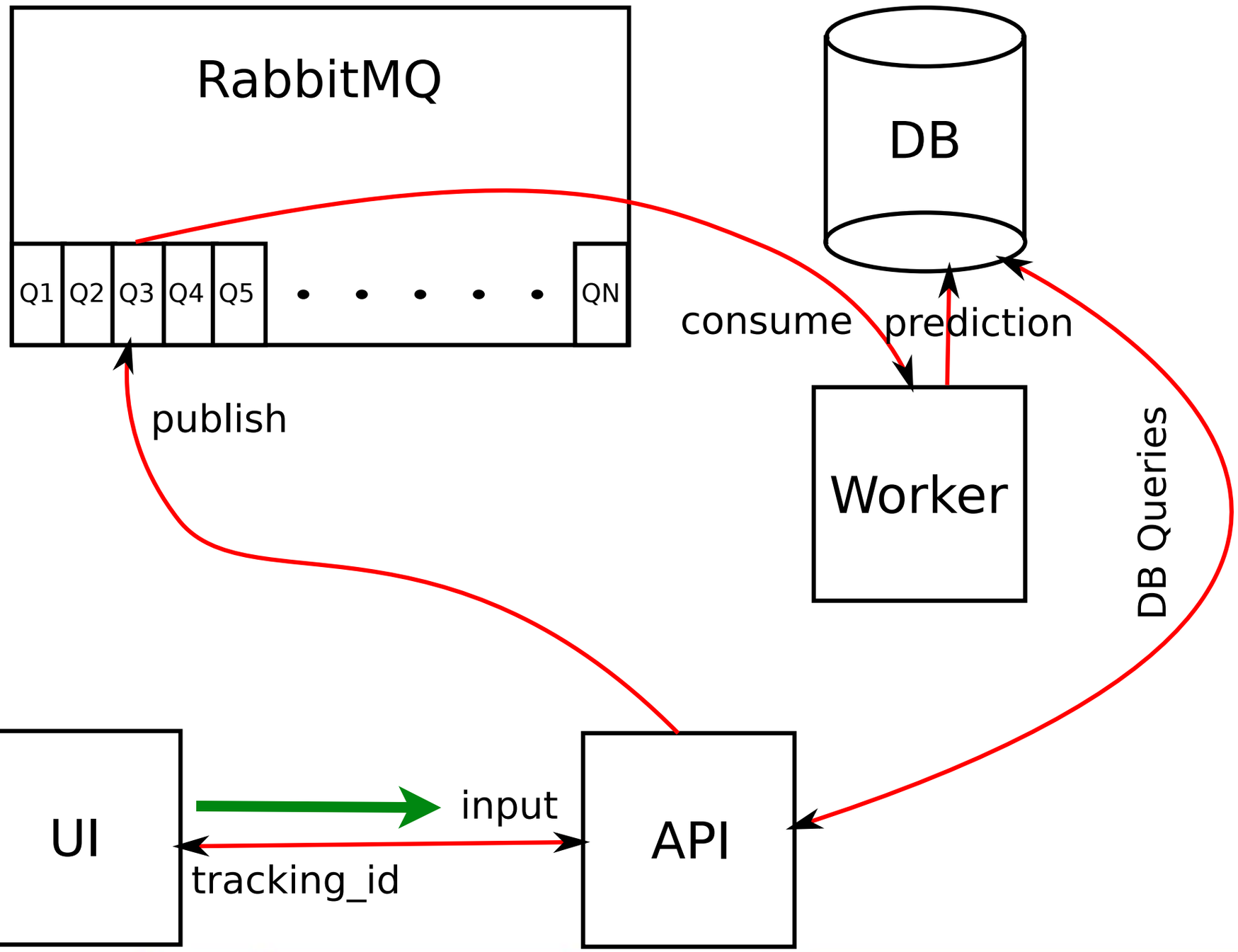
The Design


RabbitMQ
Q1
Q2
Q3
Q4
Q5
QN
DB
consume
prediction
publish
Worker
DB Queries
UI
input
tracking_id
API

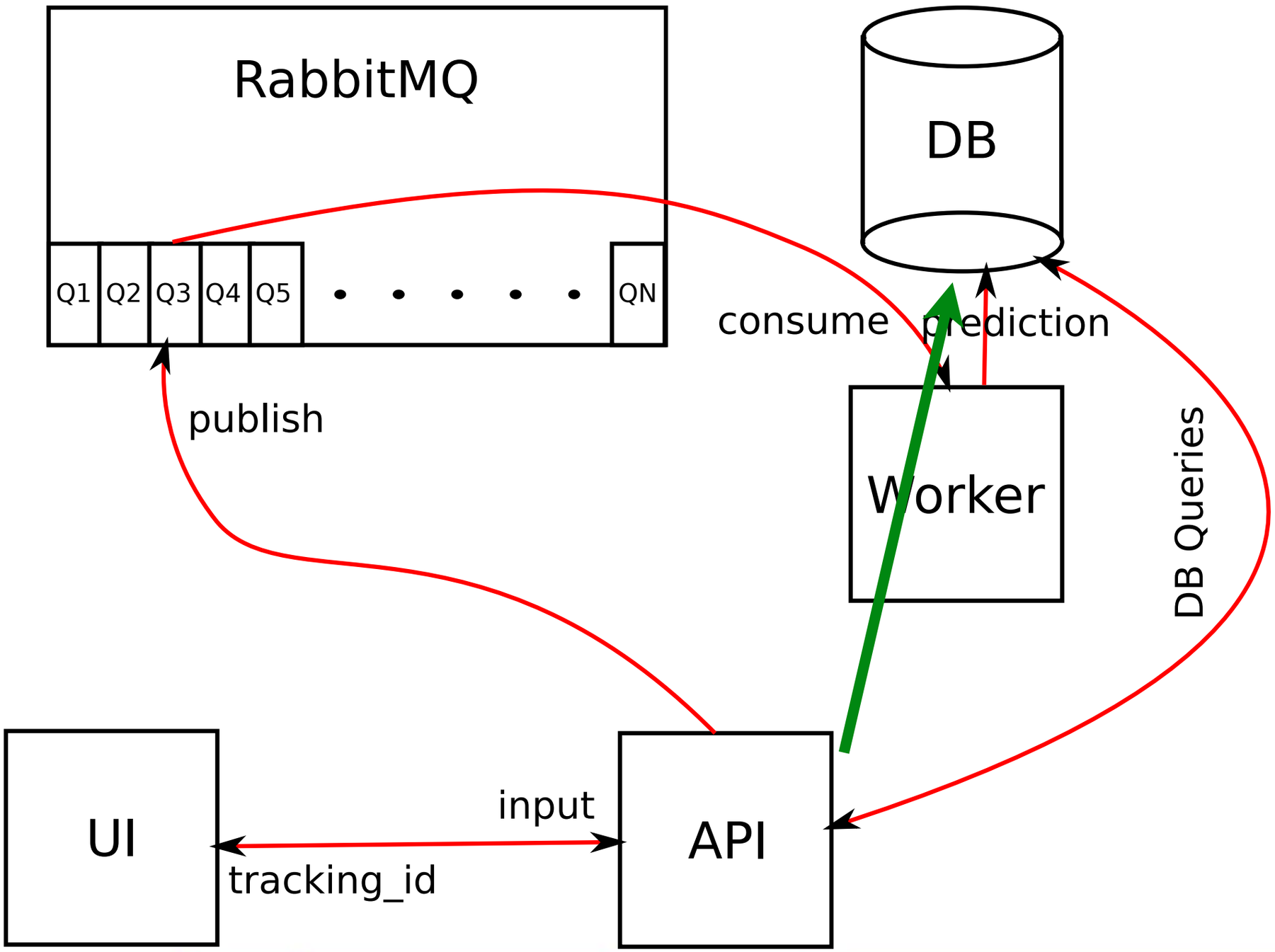
The Design


RabbitMQ
Q1
Q2
Q3
Q4
Q5
QN
DB
consume
prediction
publish
Worker
DB Queries
UI
input
tracking_id
API

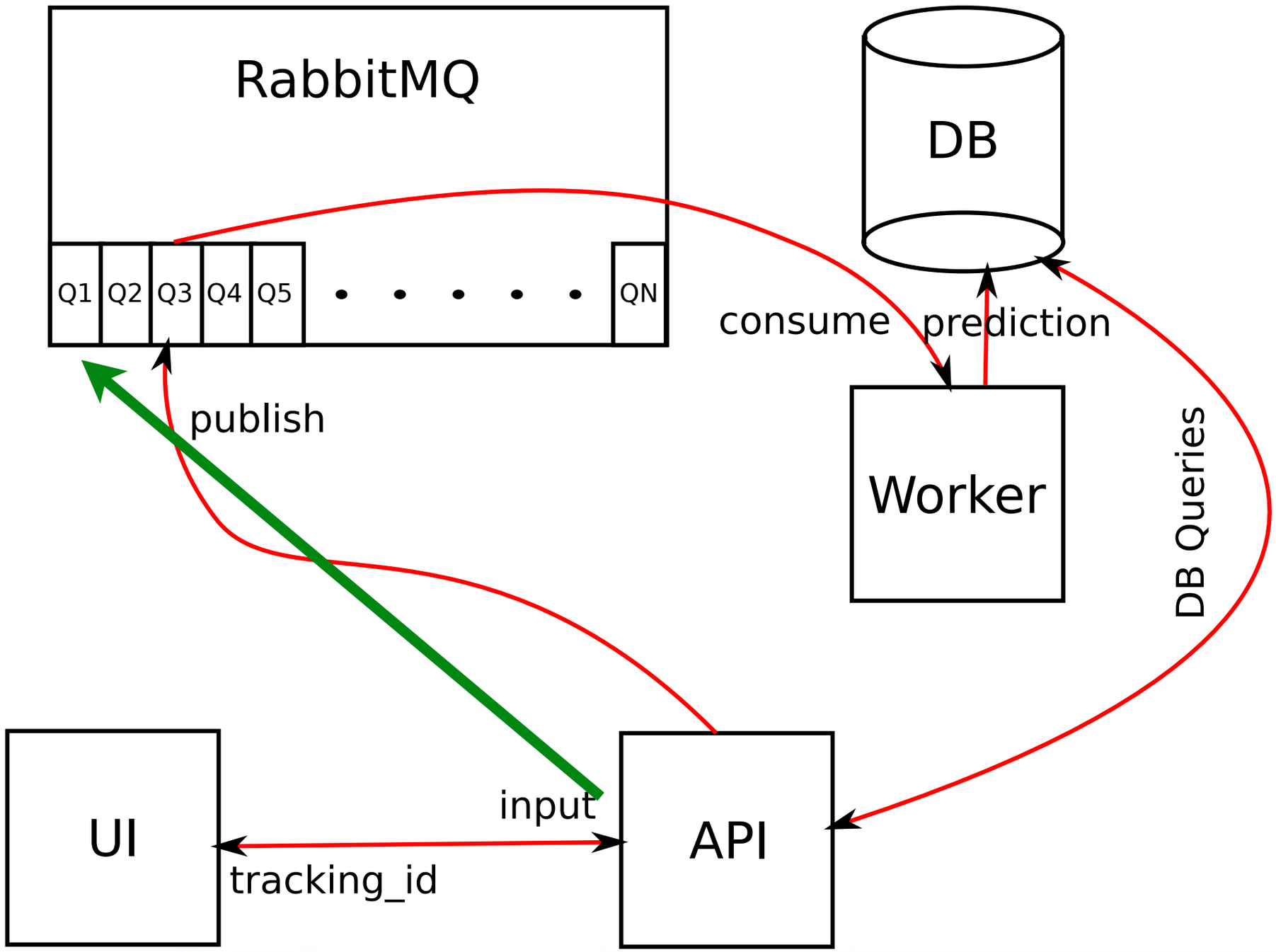
The Design


RabbitMQ
Q1
Q2
Q3
Q4
Q5
QN
DB
consume
prediction
publish
Worker
DB Queries
UI
input
tracking_id
API

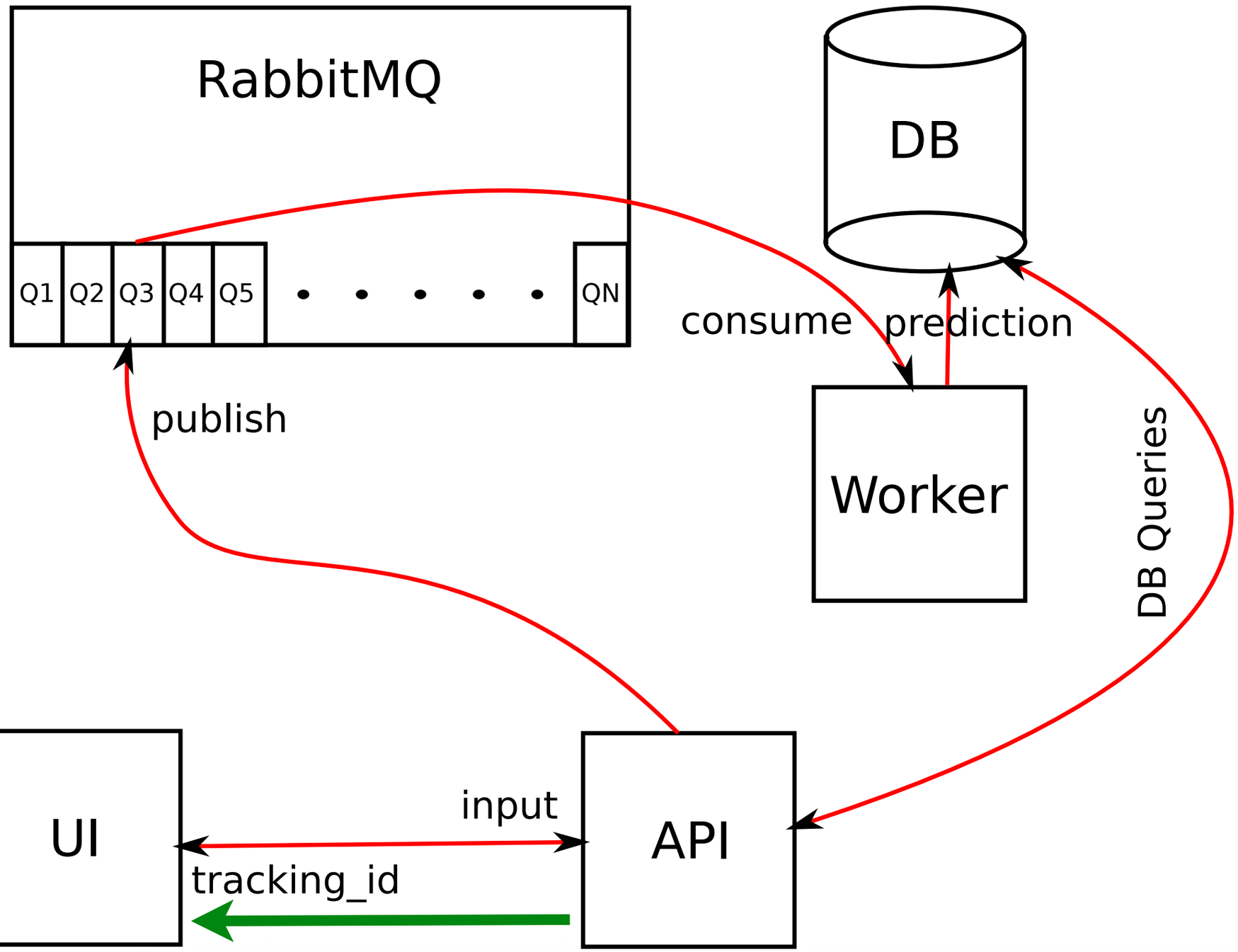
The Design


RabbitMQ
Q1
Q2
Q3
Q4
Q5
QN
DB
consume
prediction
publish
Worker
DB Queries
UI
input
tracking_id
API

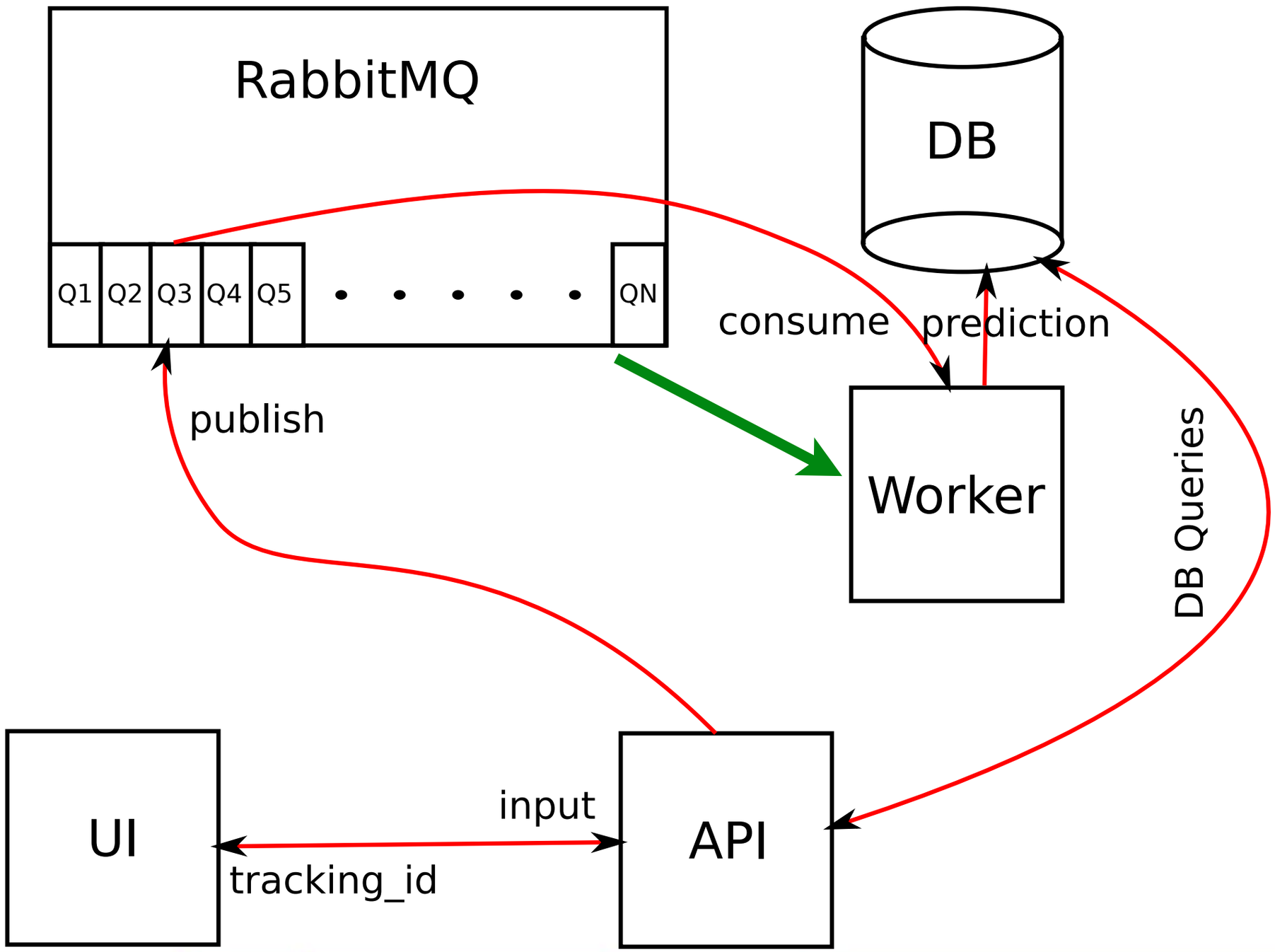
The Design


RabbitMQ
Q1
Q2
Q3
Q4
Q5
QN
DB
consume
prediction
publish
Worker
DB Queries
input
UI
API
tracking_id

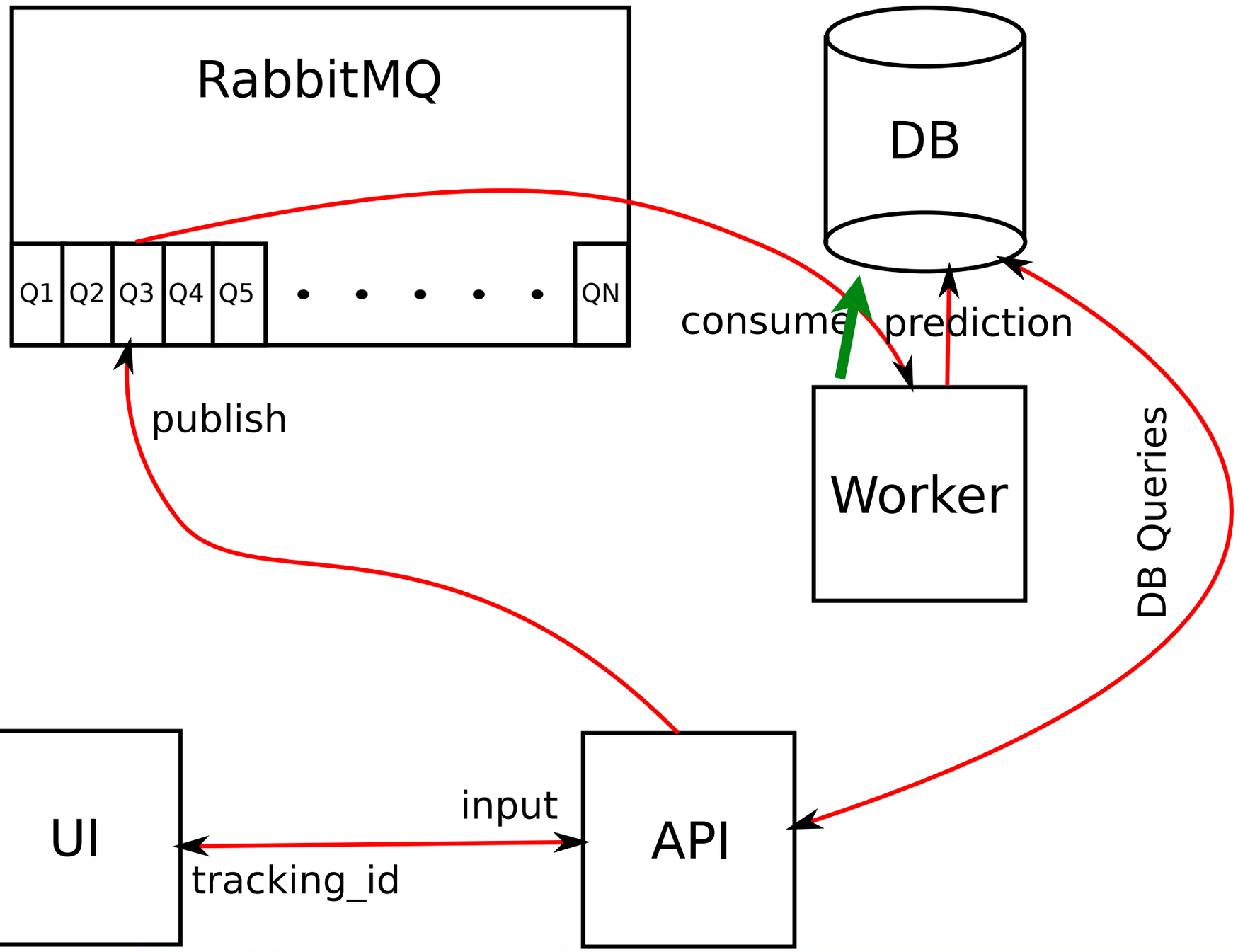
The Design


RabbitMQ
Q1
Q2
Q3
Q4
Q5
QN
DB
consume
prediction
publish
Worker
DB Queries
UI
input
tracking_id
API

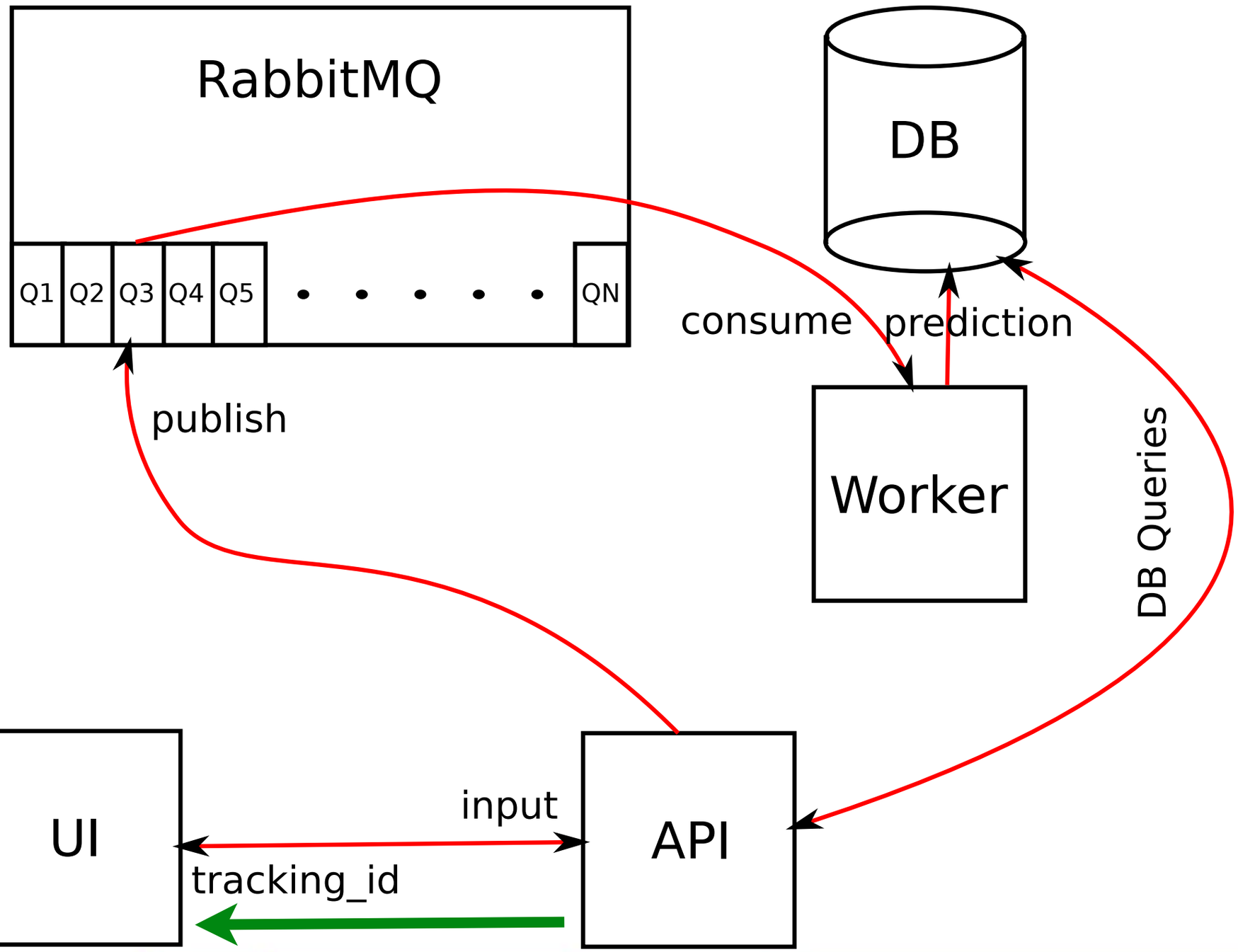
The Design


RabbitMQ
Q1
Q2
Q3
Q4
Q5
QN
DB
consume
prediction
publish
Worker
DB Queries
UI
input
tracking_id
API

The Design


RabbitMQ
Q1
Q2
Q3
Q4
Q5
QN
DB
consume
prediction
publish
Worker
DB Queries
UI
input
tracking_id
API

The Design


RabbitMQ
Q1
Q2
Q3
Q4
Q5
QN
DB
consume
prediction
publish
Worker
DB Queries
UI
input
tracking_id
API

Demo Time!

Language Prediction

Input

Adolf II av Holstein
Adolf II av Holstein, född 1128, död 6 juli 1164 (stupad vid Verchen i Demmin), begravd i Minden, var greve av Holstein 1131–1164. Son till greve Adolf I av Holstein (död 1131) och Hildewa.
Biografi[redigera]

Adolf II efterträdde fadern i Schauenburg och Holstein-Wagrien. I flera år låg dock Wagrien under kontroll av Pribislaw av Mecklenburg. I tyska tronkriget stod Adolf på welfisk sida, och han var 1138–1142 förjagad av Albrekt Björnen då Adolf vägrade erkänna denne som sachsisk hertig. Som ny greve i Holstein och Stormarn insatte Albrekt då Heinrich von Badwide, men denne kunde Adolf senare driva ut med welfisk hjälp. 1143 återfick Adolf av Henrik Lejonet sitt tidigare grevskap mot en stor summa pengar, och detta år förenades landskapet Wagrien slutgiltigt med Holstein.
Adolf grundade 1134 Segeberg vars borg senare brändes ned av den av Adolf på flykt fördrivne Heinrich von Badwide. Borgen återuppbyggdes och blev Adolfs viktigaste stödjepunkt. 1143 grundade Adolf även Alt-Lübeck som förstördes 1147 av furst Niklot av obotriterna. Området överlämnades slutligen till Henrik Lejonet vilken 1158 nygrundade Lübeck.

Submit

Result

Swedish Svenska

Scaling

Realtime vs Batch

Monitoring

Load

Re-verify

Re-verify

Thank You!

API & Consumer: Kelly O'Brien (linkedin.com/in/kellyobie)

UI: Matt Parke (ordinaryrobot.com)

Classifier: Michael Becker (github.com/mdbecker)

Images: Wikipedia; flickr.com users: kohsa, spenceyc, statefarm, klara, rh2ox, nasahqphoto, frickfrack

My Info

Twitter: @beckerfuffle

Blog: beckerfuffle.com

These slides and more @ github.com/mdbecker

Demo time!

Language Prediction

Input

Adolf II av Holstein
Adolf II av Holstein, född 1128, död 6 juli 1164 (stupad vid Verchen i Demmin), begravd i Minden, var greve av Holstein 1131–1164. Son till greve Adolf I av Holstein (död 1131) och Hildewa.
Biografi[redigera]

Adolf II efterträdde fadern i Schauenburg och Holstein-Wagrien. I flera år låg dock Wagrien under kontroll av Pribislaw av Mecklenburg. I tyska tronkriget stod Adolf på welfisk sida, och han var 1138–1142 förjagad av Albrekt Björnen då Adolf vägrade erkänna denne som sachsisk hertig. Som ny greve i Holstein och Stormarn insatte Albrekt då Heinrich von Badwide, men denne kunde Adolf senare driva ut med welfisk hjälp. 1143 återfick Adolf av Henrik Lejonet sitt tidigare grevskap mot en stor summa pengar, och detta år förenades landskapet Wagrien slutgiltigt med Holstein.
Adolf grundade 1134 Segeberg vars borg senare brändes ned av den av Adolf på flykt fördrivne Heinrich von Badwide. Borgen återuppbyggdes och blev Adolfs viktigaste stödjepunkt. 1143 grundade Adolf även Alt-Lübeck som förstördes 1147 av furst Niklot av obotriterna. Området överlämnades slutligen till Henrik Lejonet vilken 1158 nygrundade Lübeck.

Submit

Result

Swedish Svenska

Demo time!

Language Prediction

Input

Manhattanprosjektet (engelsk: Manhattan Project) var et forsknings- og utviklingsprogram, som under amerikansk ledelse og med deltakelse av Storbritannia og Canada førte til fremstillingen av de første atombombene under andre verdenskrig. Fra 1942 til 1946 ble prosjektet ledet av generalmajor Leslie Groves fra den amerikanske hærens ingeniørkorps (US Army Corps of Engineers). Hærens del av prosjektet fikk betegnelsen Manhattan District. Manhattan avløste etter hvert det offisielle kodenavnet Development of Substitute Materials som betegnelse for hele prosjektet. Underveis slukte Manhattanprosjektet også den tidligere britiske motparten, kalt Tube Alloys. Blant fysikerne som tok del i Manhattanprosjektet var Albert Einstein, Edward Teller og danske Niels Bohr.
Manhattanprosjektet begynte i det små i 1938, men det vokste raskt og tilsammen beskjeftiget det over 130 000 personer og kostet nesten 2 milliarder dollar (tilsvarende ca. 150 milliarder i 2012[1]). Over 90 % av pengene gikk til byggingen av fabrikker og produksjonen av spaltbart materiale, mens under 10 % gikk til selve utviklingen av våpnene. Forskning og utvikling foregikk på mer enn 30 forskjellige steder rundt om i USA, Storbritannia og Canada, og noen av disse var hemmelige.
Det ble bygget to typer atombomber under andre verdenskrig. En forholdsvis enkel uranbombe som benyttet uran-235, som er en relativt sjelden uranisotop som kun utgjør 0,7 % av naturlig

Submit

Result

Norwegian (Bokmål) Norsk (Bokmål)

Demo time!

Language Prediction

Input

مره أخرى لأنه ينفذ أوامر القادة.
ولد (بول تيبيتس) في عام 1915 م واسم والدته "اينولا جاي اشتهر تيبيتس بانه أفضل طياري الجيش الأمريكي وقتها وكان تيبيتس، الذي اشتهر بانه أفضل طياري الجيش الأمريكي وقتها، وفي عام 1945 م ترقي إلي رتبة كولونيل وهي تعادل رتبة (عقيد)في سلاح الجو الأمريكي في 6 -8- 1945 قاد الطائرة الأمريكية التي ألقت القنبلة الذرية على هيروشيما في اليابان وهي قنبلة تحتوي على 60 كيلوغراما (130 رطلا) من مادة اليورانيوم ـ 235 هيروشيما هي مدينة في اليابان، تقع في جزيرة "هونشو"، وتشرف على "خليج هيروشيما".وكان تعداد سكانها عام 1945 م وأما الطائرة التي تحمل القنبلة فكانت قاذفة من (Little Boy 350,000 نسمة وكان اسم الطائرة الكودي (بالشفرة السرية) هو (الولد الصغير النوع (بي 29) وفكان بول تيبيتس أول من اختبر هذه الطائرة وأطلق عليها اسم أمه "اينولا جاي". وتقرر تنفيذ مهمة قصف هيروشيما يوم 6-8- 1945 م حيث كان الطقس مواتيا لتنفيذ المهمة الخطيرة بعد أيام من السحب التي تجمعت فوق هيروشيما، فانطلق بول تيبيتس بطائرته وهي تحمل القنبلة الذرية من قاعدة «نورث فيلد» في جزيرة تنيان، غرب المحيط الهادئ، مصحوباً بطائرتين أخرين. وقبل القصف بساعة اكتشف نظام الإنذار المبكر الياباني دخول الطائرات للمجال الجوي الياباني فنبه السلطات في كبرى المدن بما فيها هيروشيما. لكن بول تيبيتس كان في طريقه للمدينة المستهدفة يقود الطائرة القاذفة، واستطاع أن يتهرب من الدفاعات اليابانية ليصل مدينة هيروشيما وحوالي الساعة الثامنة صباحا تمكنت أجهزة الرادار في هيروشيما من تحديد الطائرات الأميركية لكن المسؤولين العسكريين قرروا ان عددها الصغير لا يستدعي التصدي لها بطائرات مضادة على ضوء سياستهم الرامية لتوفير وقود الطائرات0 وفي تمام الساعة الثامنة والربع قصف بول تيبيتس القنبلة الرهيبة من طائرته «بي 29» على

Submit

Result

Arabic العربية