DataStax

Developers

NLP as an API

Text classification using Keras, FastAPI and NoSQL

Based on original content by CodingEnterpreneurs

01

Housekeeping

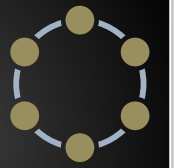
Live and Hands-on

02

Our Goal

Use case and Technologies

03



Database Setup

Data model & Astra DB

04

AI

Train a text classifier

05

API

Bring to production

06

What's next?

Quiz, Homework, Agenda

Housekeeping

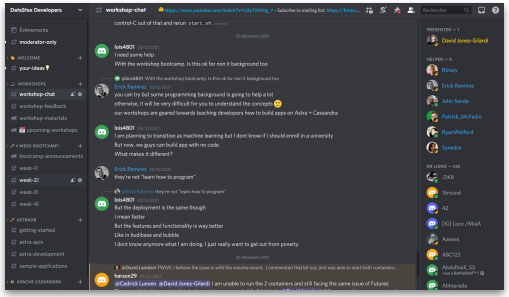
Live and Hands-on

Livestream: youtube.com/DataStaxDevs

YouTube

Twitch

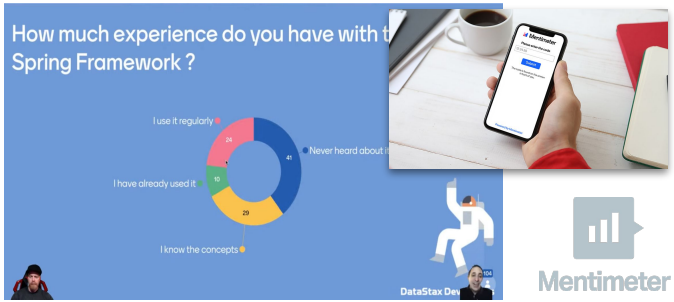
Questions: https://dtsx.io/discord



YouTube

Discord

Games and quizzes: menti.com

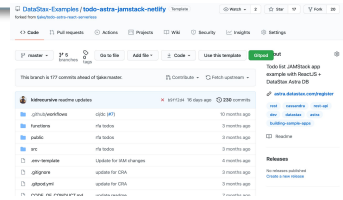


Mentimeter

Live Sessions

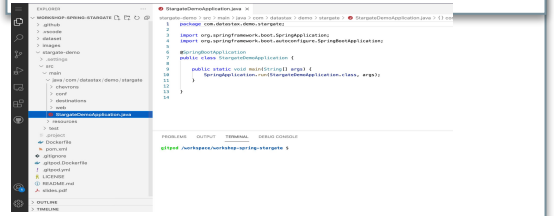
Nothing to install !

Source code + exercises + slides



GitHub

IDE



Gitpod

Database + GraphQL + PlayGround

DataStax
Astra DB

CodingEnterpreneurs

Youtube & blog for more

Hands-On Housekeeping

01

Housekeeping

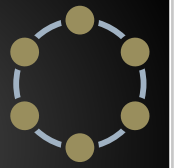
Live and Hands-on

02

Our Goal

Use case and Technologies

03



Database Setup

Data model & Astra DB

04

AI

Train a text classifier

05

API

Bring to production

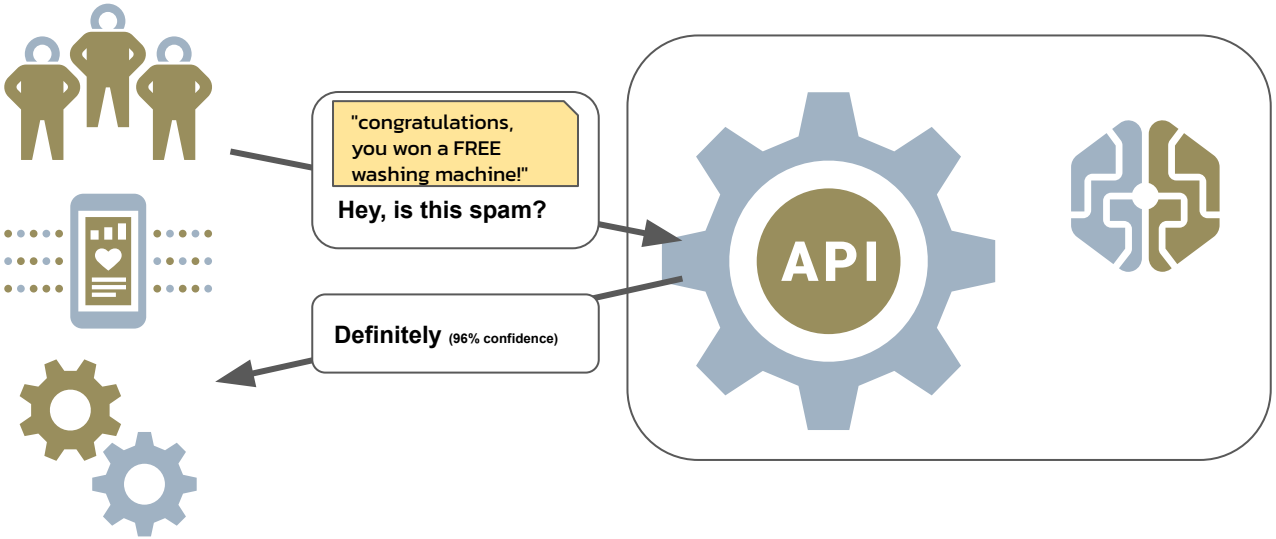
06

What's next?

Quiz, Homework, Agenda

Our Goal

Use cases and technologies

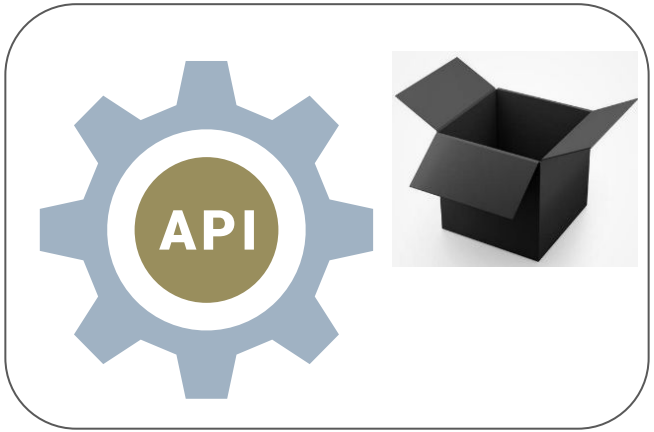

"congratulations, you won a FREE washing machine!"
Hey, is this spam?
Definitely (96% confidence)
API

What we want

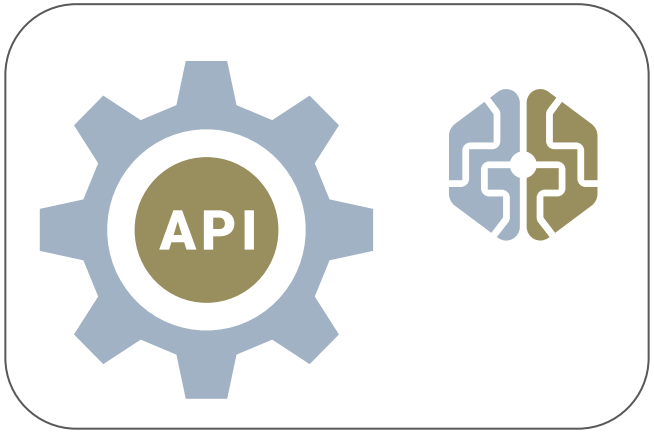
The API sees the AI part as a black box

API = concerned with stability & performance

Engineer Stuff™



Traditional viewpoint

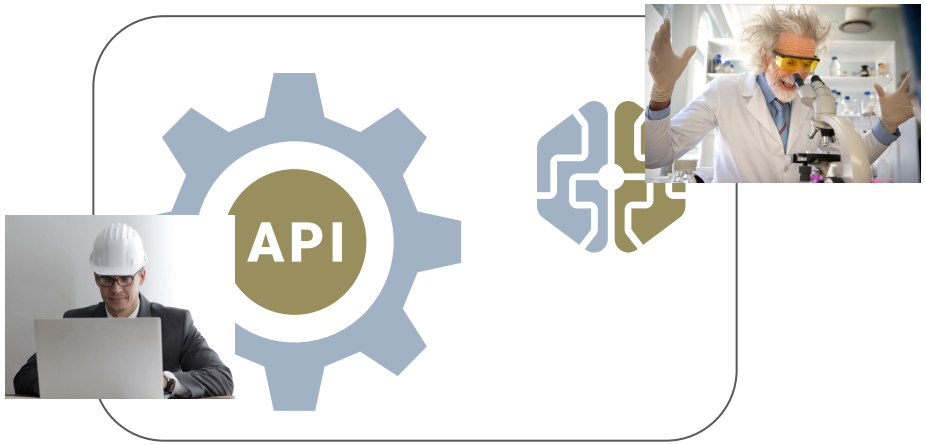


ML components, often messy
(no offence meant)

(hardly reproducible) treasures
hidden away

A "science" proper
(experimentation, prototyping…)

Traditional viewpoint

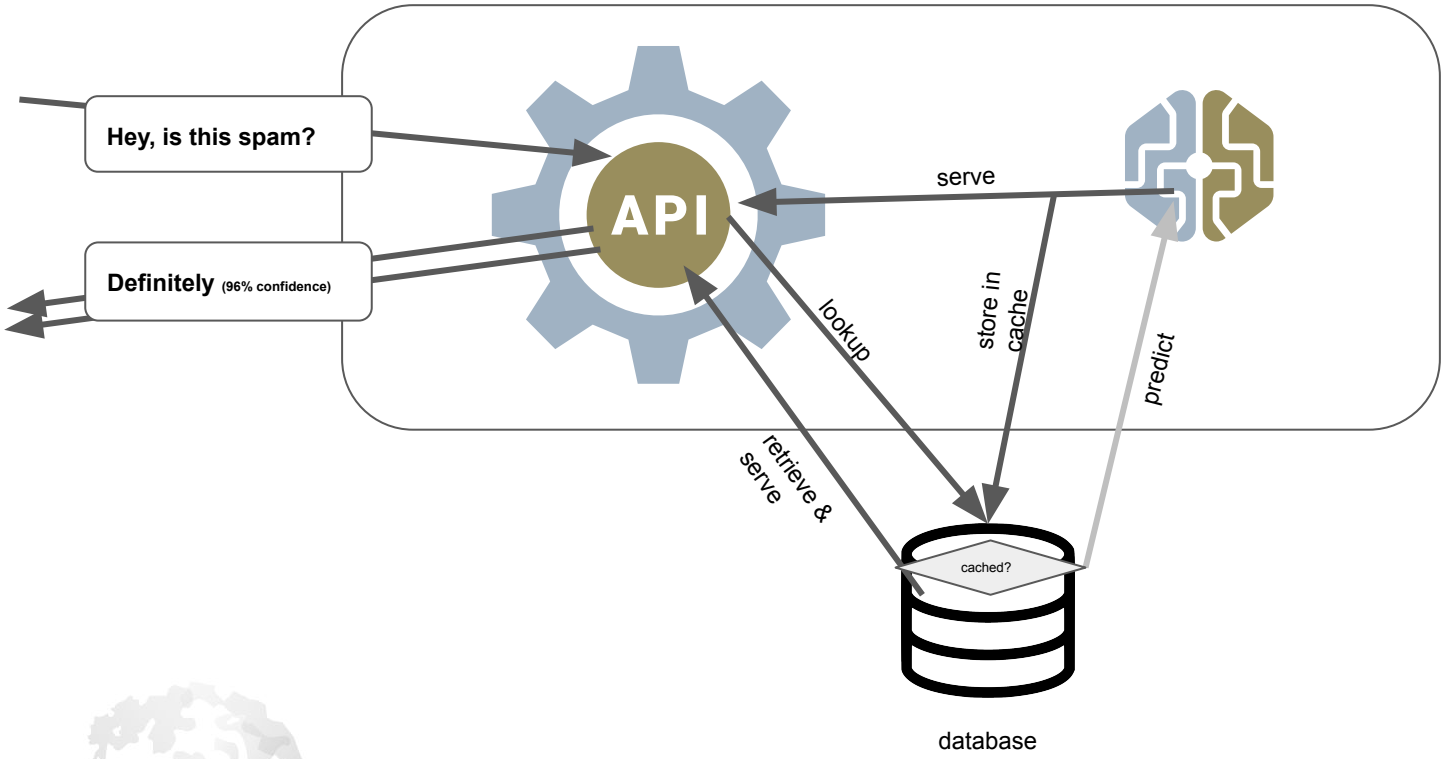


Properly exposing ML elements
in production is a key skill

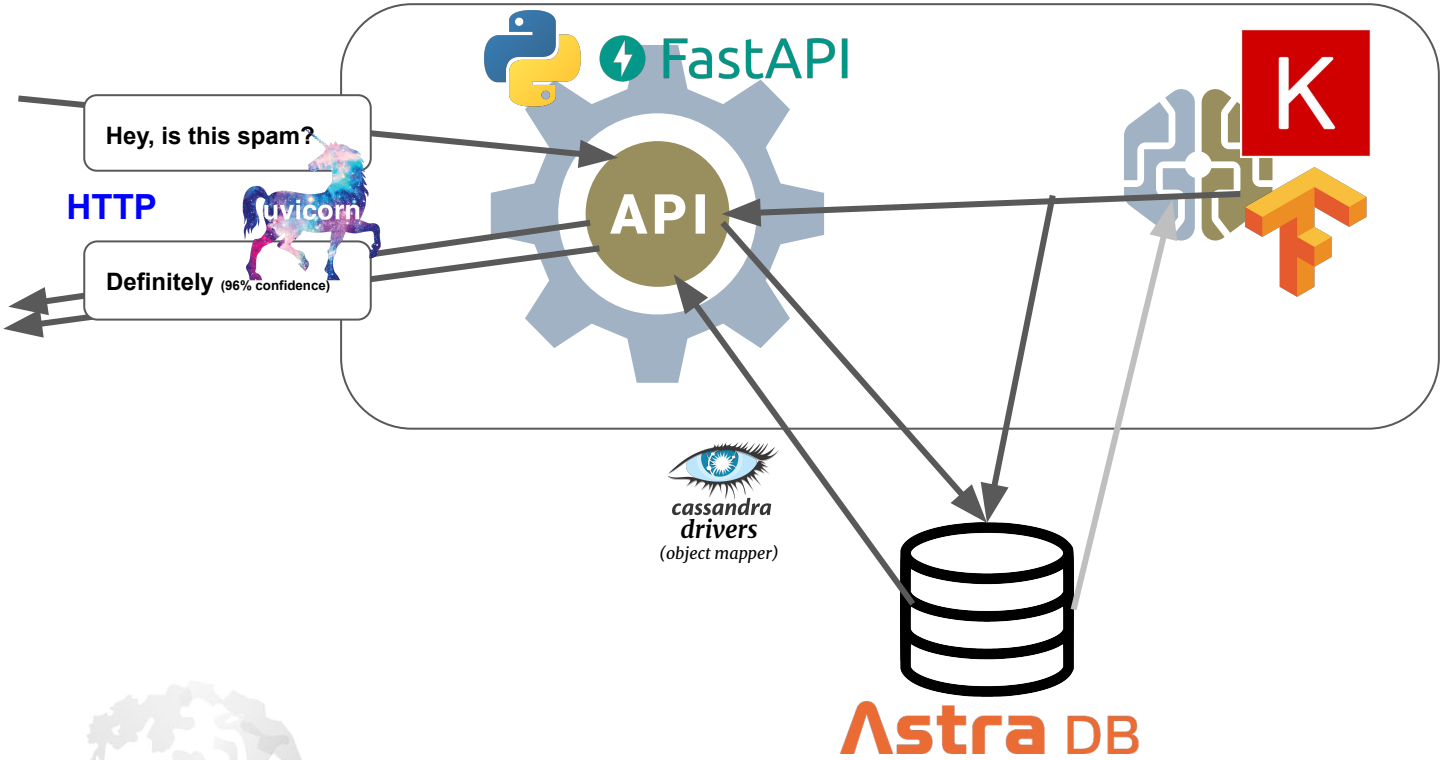
Still, two clearly separate phases:

- create the classifier (design, train)
- bring it to production

Two in one



Architecture sketch


Hey, is this spam?
HTTP
uvicorn
Definitely (96% confidence)
FastAPI
API
K
cassandra
drivers
(object mapper)
Astra DB

Tech stack

01

Housekeeping

Live and Hands-on

02

Our Goal

Use case and Technologies

03

Database Setup

Data model & Astra DB

04

AI

Train a text classifier

05

API

Bring to production

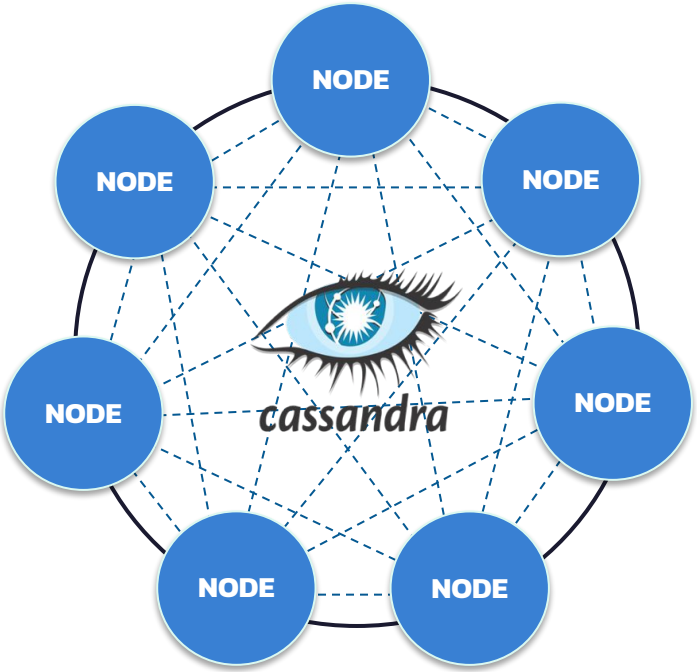
06

What's next?

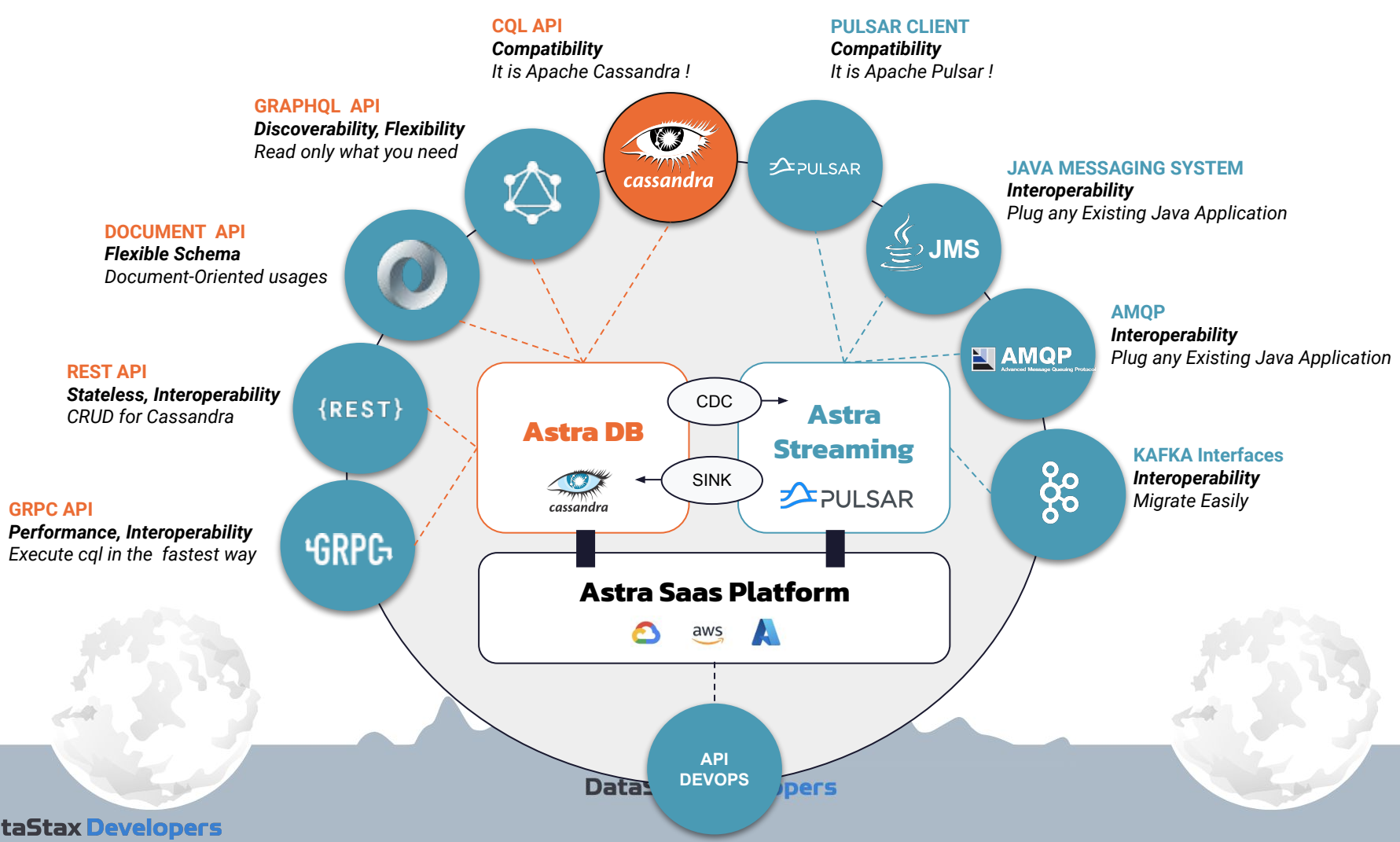
Quiz, Homework, Agenda

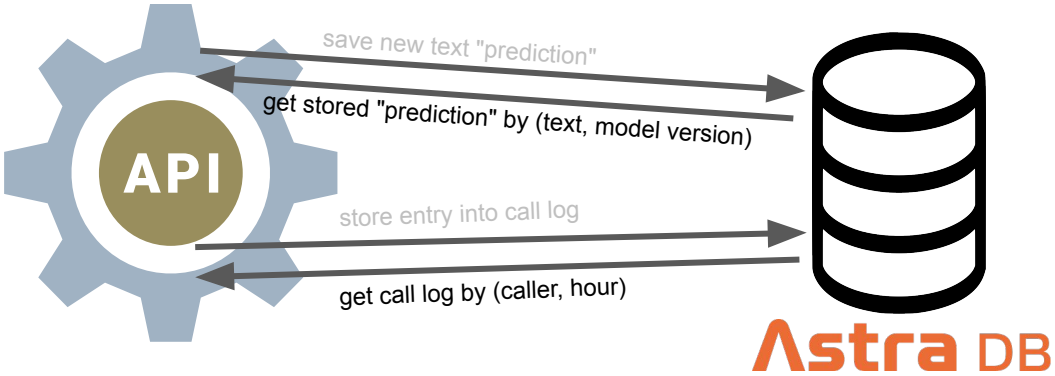
Database setup

Data model & Astra DB



- Big Data Ready
- Read / Write Performance
- Linear Scalability
- Highest Availability
- Self-Healing and Automation
- Geographical Distribution
- Platform Agnostic
- Vendor Independent

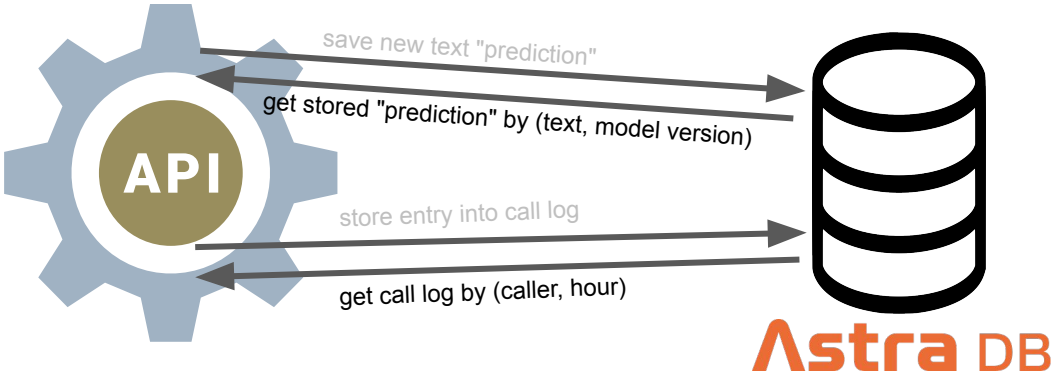

CQL API
Compatibility
It is Apache Cassandra !
PULSAR CLIENT
Compatibility
It is Apache Pulsar !
GRAPHQL API
Discoverability, Flexibility
Read only what you need
JAVA MESSAGING SYSTEM
Interoperability
Plug any Existing Java Application
DOCUMENT API
Flexible Schema
Document-Oriented usages
AMQP
Interoperability
Plug any Existing Java Application
REST API
Stateless, Interoperability
CRUD for Cassandra
KAFKA Interfaces
Interoperability
Migrate Easily
GRPC API
Performance, Interoperability
Execute cql in the fastest way
cassandra
PULSAR
JMS
AMQP
{REST}
GRPC
Astra DB
cassandra
CDC
SINK
Astra Streaming
PULSAR
Astra Saas Platform
aws
API
DEVOPS



Data modeling, the Cassandra way:

"design the table after the query"

(also: tables are partitioned!)



spam_cache_items

model_version	text	K
input	text	K
confidence	float	
prediction_map	map<text,float>	
result	text	
stored_at	timeuuid	

spam_calls_per_caller

caller_id	text	K
called_hour	timestamp	K
called_at	timeuuid	C↑
input_text	text	

"partition key" (rows stored together)

"primary key" (row uniqueness)

"Chebotko diagrams"

Data modeling

(object mapper)

```python
class SpamCallItem(Model):
    __table_name__ = 'spam_calls_per_caller'
    __keyspace__ = ASTRA_DB_KEYSPACE
    __connection__ = 'my-astra-session'
    caller_id = columns.Text(primary_key=True, par
    called_hour = columns.DateTime(primary_key=Tru
    called_at = columns.TimeUUID(primary_key=True,
    input = columns.Text()
```

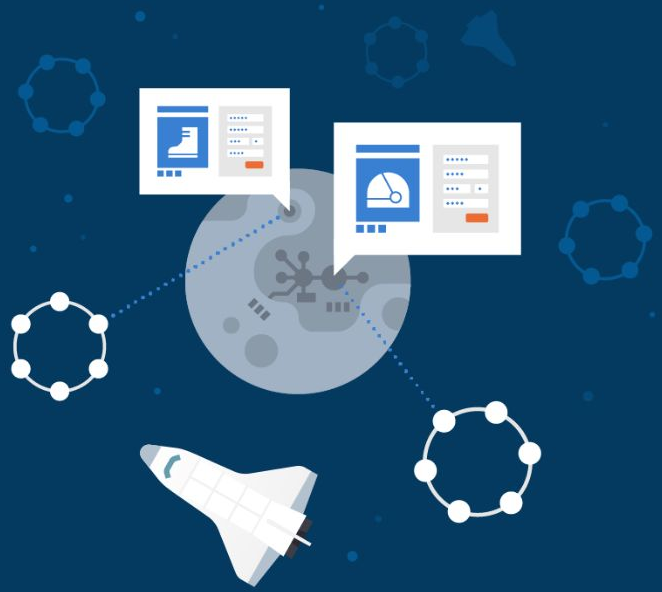
Tables are created automatically…

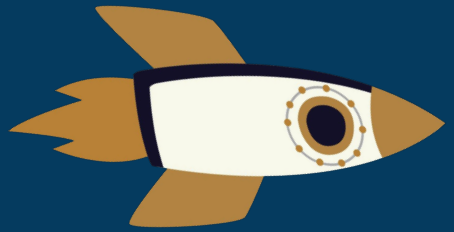
```python
query = SpamCallItem.objects().filter(
    caller_id=caller_id,
    called_hour=called_hour,
)
for item in query:
    yield item
```

```python
#
# Database
logging.info('     DB initialization')
DBSession = initSession()
sync_table(SpamCacheItem)
sync_table(SpamCallItem)
logging.info('     API Startup completed.')
```

```python
called_hour = getThisHour()
for input in inputs:
    SpamCallItem.create(
        caller_id=caller_id,
        called_hour=called_hour,
        input=input,
    )
```

DB I/O in practice





Hands-on (!github)

DB SETUP

- ✅ Create (or resume) your Database
- ✅ Create a token (if needed)
- ✅ Download secure-connect-bundle

01

Housekeeping

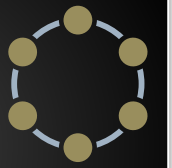
Live and Hands-on

02

Our Goal

Use case and Technologies

03



Database Setup

Data model & Astra DB

04

AI

Train a text classifier

05

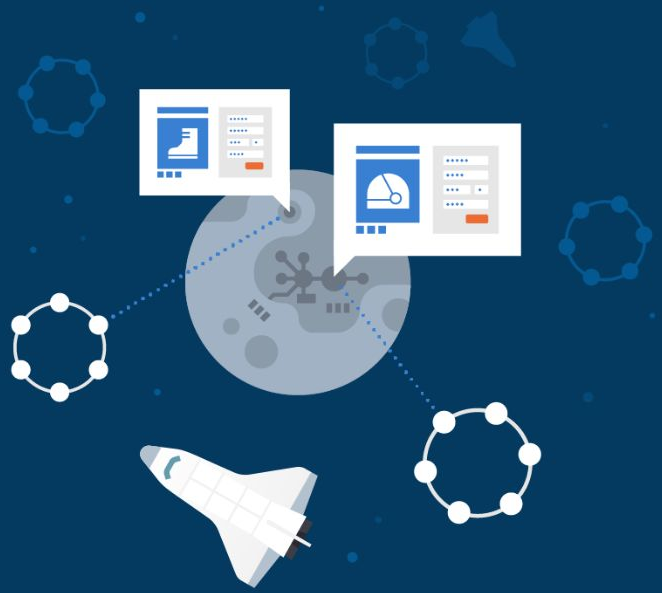
API

Bring to production

06

What's next?

Quiz, Homework, Agenda

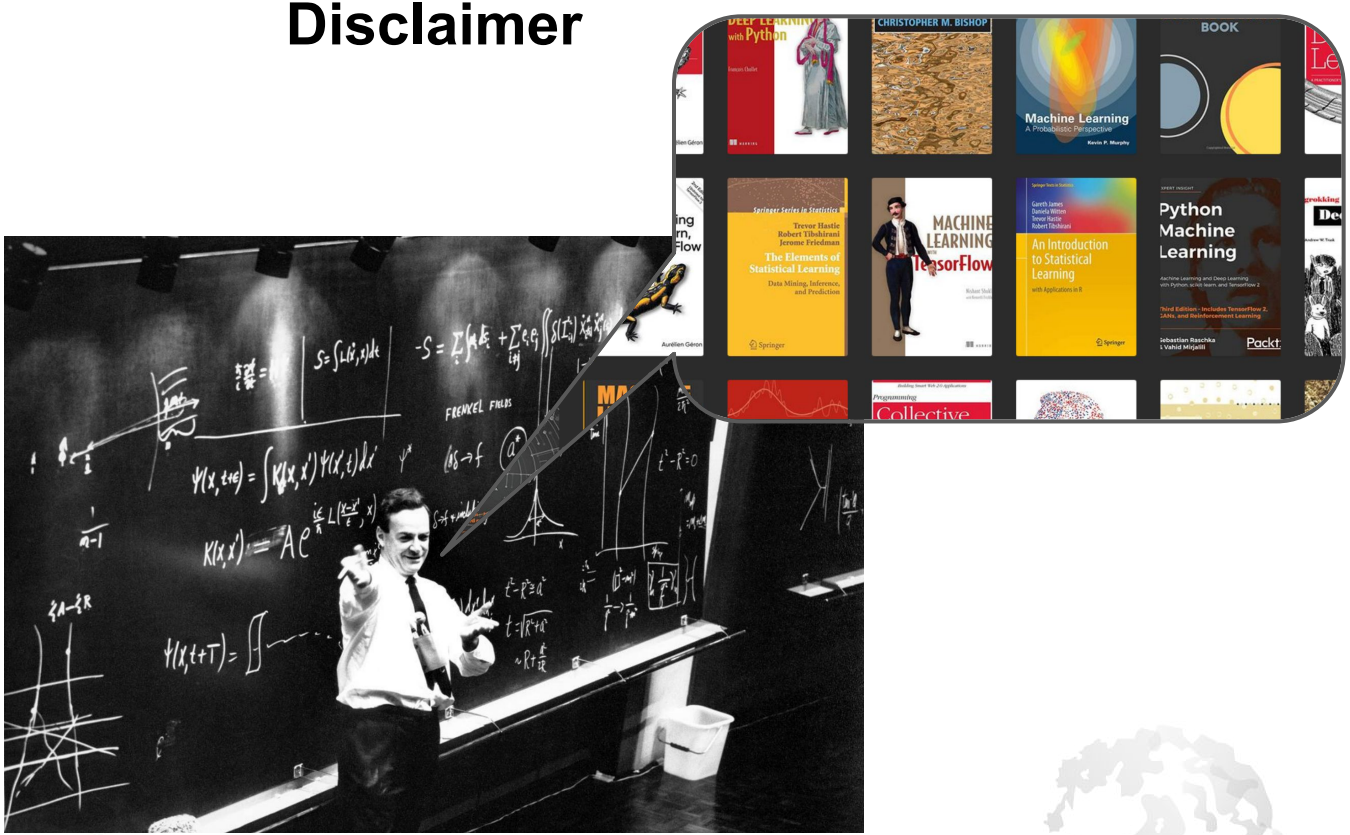


AI

Train a text classifier

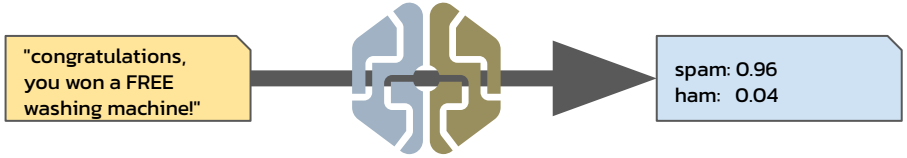
Disclaimer

What this section is **not**:



"ML: LSTM RNN for NLP"

This is what we want:

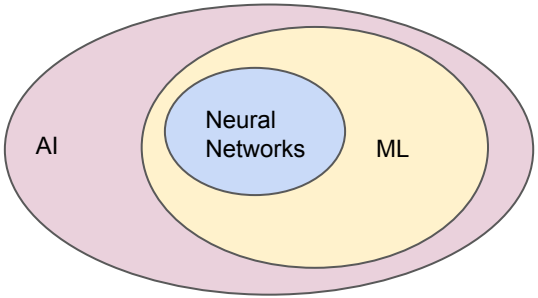


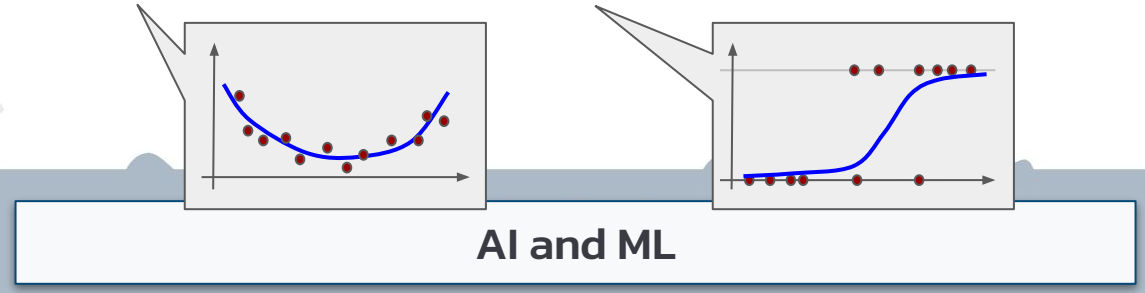
Machine Learning = *algorithms that improve by being fed data, without being explicitly instructed what to do.*

It's essentially statistical inference (with superpowers).

Lots of **math** involved (linear algebra, calculus, prob./statistics).
Nowadays accessible as neatly-packaged tools (good for us!)

Simple examples of ML: least-squares fits, logistic regressions.





"Blank slate" model

Training
(i.e. tune parameters)

Ready to predict !

(classify, recognize objects in images, do speech-to-text, …)

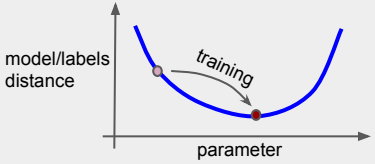
Choice of the model
(also "hyperparameters")

Labeled training data

X	Y
Go until jurong poi	ham
Ok lar... Joking wi	ham
Free entry in 2 a wk	spam
U dun say so early h	ham
Nah I don't think he	ham
FreeMsg Hey there da	spam
Even my brother is n	ham
As per your request	ham
WINNER!! As a valued	spam
Had your mobile 11 m	spam

"Training":

tuning model parameters to make predictions as close to labeled data as possible *(more later)*



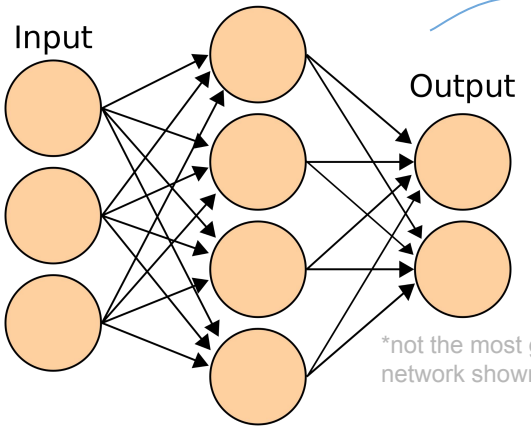
Supervised learning

Many algorithms and structures for ML:

linear/logistic regression, support-vector-machines, decision trees/random forests, gradient boosting, …

Focus on **Neural Networks**

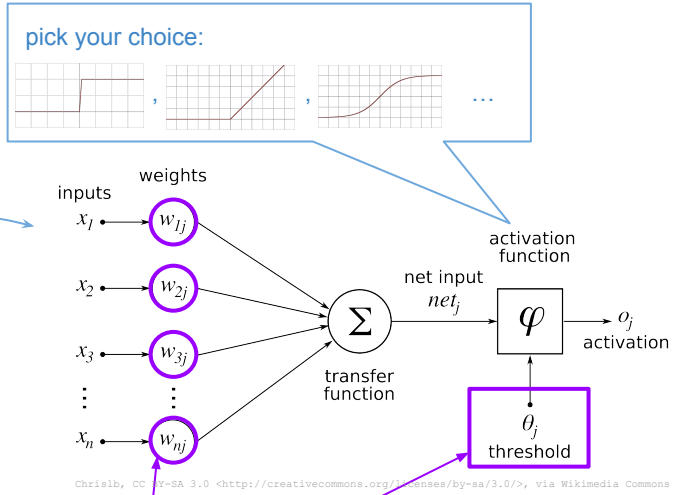
"deep" networks have several hidden layers



*not the most general network shown here

Input (text, image, …) is encoded and fed here

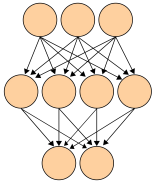
the j-th neuron looks like:



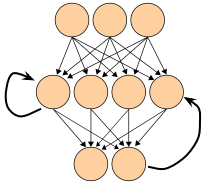
many parameters per neuron:
training gets CPU- and time-intensive

Neural Networks

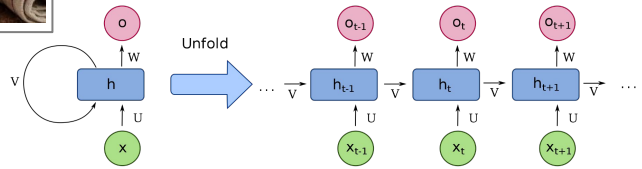
Architectures



"Feed-forward"

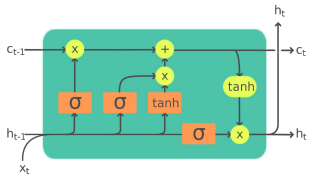


"Recurrent neural network"



has *loops*: ~ "memory"
better with sequences (time series, speech, **texts**, …)
loops "unfold" as time-steps: training gets harder

"Long-short-term-memory"



introduces explicit memory elements
overcomes the "vanishing gradient" training problem
a standard RNN to process **"input sequences"**

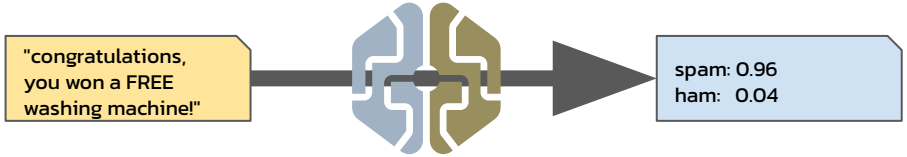
Natural language processing

A very wide field of application for ML (since … forever)

translation, summarization, classification, clustering, autocompletion, generation, chatbots, sentiment analysis, …

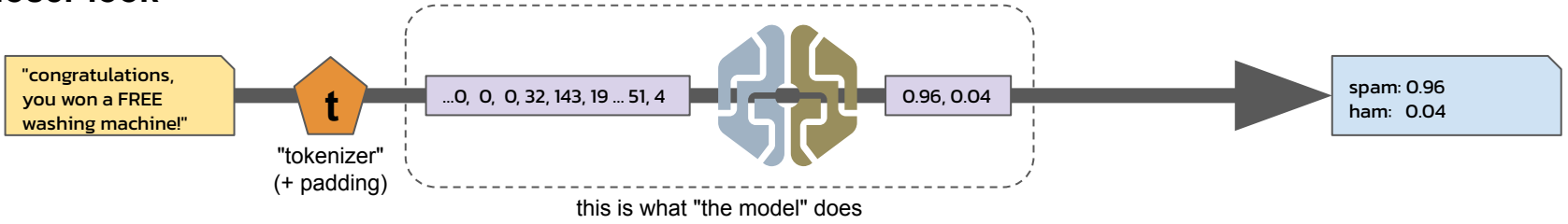
Most of the data "out there" is unstructured (e.g. **text**)

LSTM very apt at many of these tasks (memory along sequences)

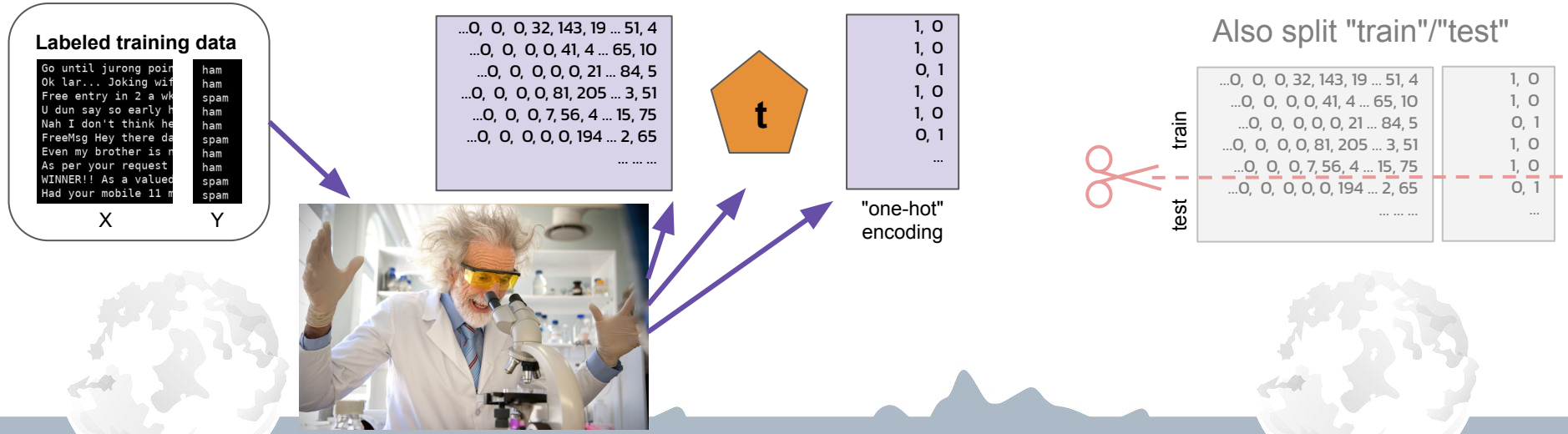


An evolving field (Word2Vec, ca. 2013; transformers e.g. BERT, ca. 2018, …)

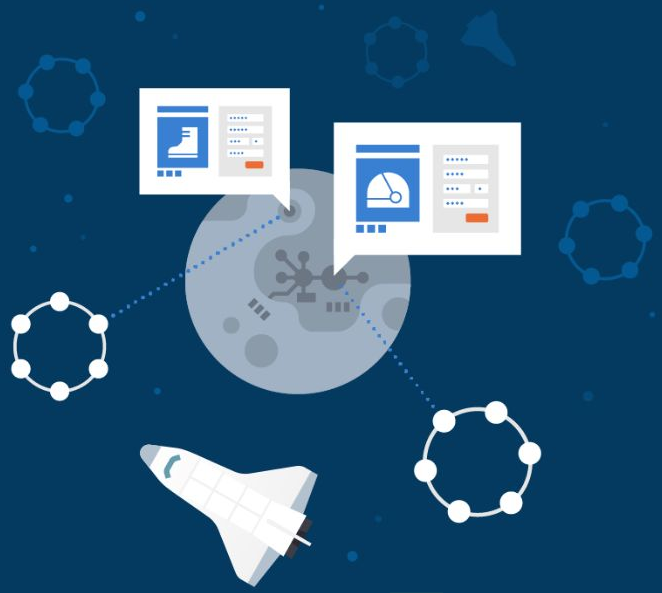
A closer look

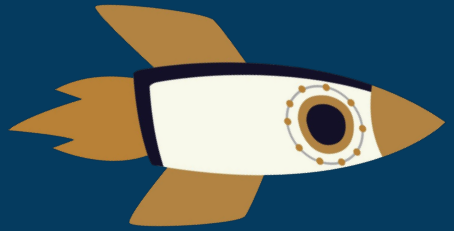


We must then *prepare the dataset* before training (including building the tokenizer):



Numeric encoding





Hands-on (!github)

TRAIN THE MODEL

- ✅ Prepare the dataset for training
- ✅ Train the NLP model
- ✅ Test the saved model

```
model = Sequential()
model.add(Embedding(maxNumWords, embedDim, 1
model.add(SpatialDropout1D(0.4))
model.add(LSTM(lstmOut, dropout=0.3, recurre
model.add(Dense(2, activation='softmax'))
model.compile(loss='categorical_crossentropy
```

A stack of layers:

one translates an input number into a vector

one randomly disables pieces of network to enhance training ("dropout")

then the LSTM (recurrent *within the layer*)

and a final ordinary layer reducing it all to a 2-component output (spam/ham)

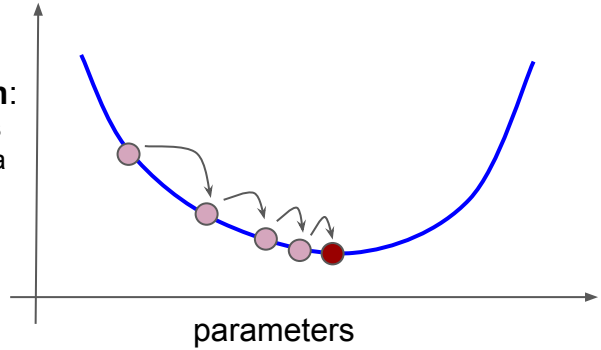
"finalize" the model: ready to train!

"Training":

(progressively) tuning model parameters to make predictions as close to labeled data as possible

loss function:
distance between provided labels and model output for training data

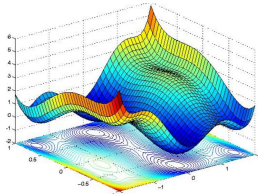
e.g. "categorical cross-entropy" (from information theory)



The reality:

lots of parameters, it's a mess!

```
=================================
Total params: 291,034
Trainable params: 291,034
Non-trainable params: 0
```



"Stochastic gradient descent" (SGD) 

Backpropagation (glorified chain rule )

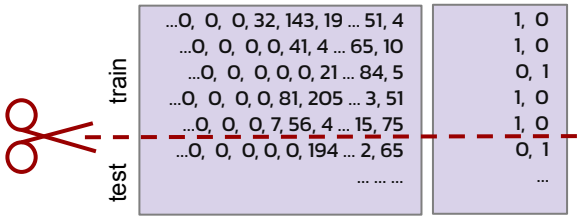
Drop-out 

...

During training ...

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

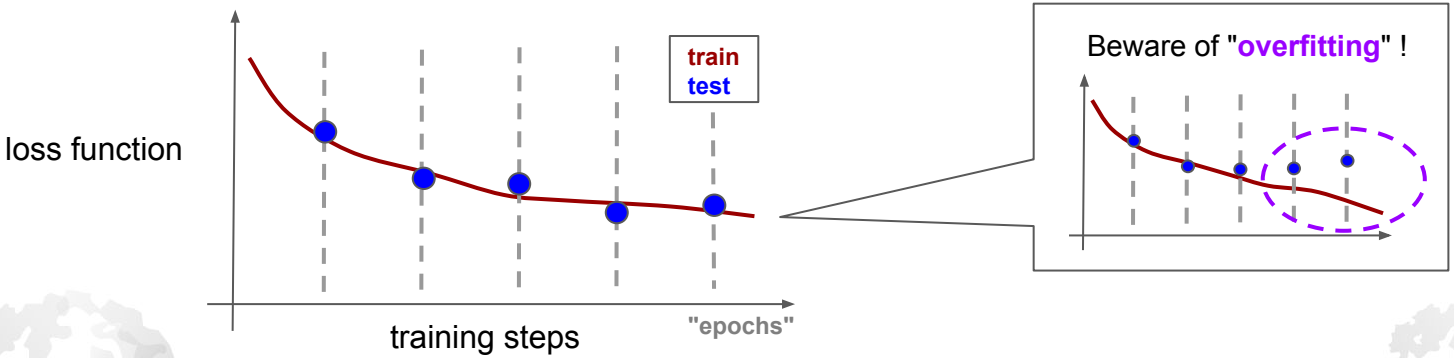
We split "train"/"test"



Train on part of input dataset,

periodically validate on *another portion* of it

(independently verify we're actually learning the right problem)



Approaching the trained model

01

Housekeeping

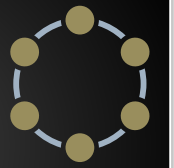
Live and Hands-on

02

Our Goal

Use case and Technologies

03



Database Setup

Data model & Astra DB

04

AI

Train a text classifier

05

API

Bring to production

06

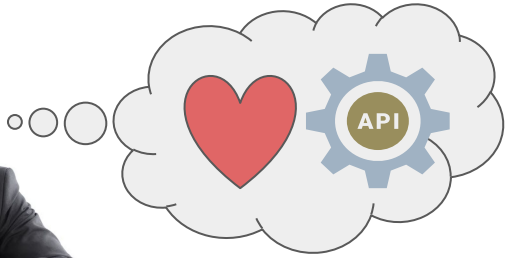
What's next?

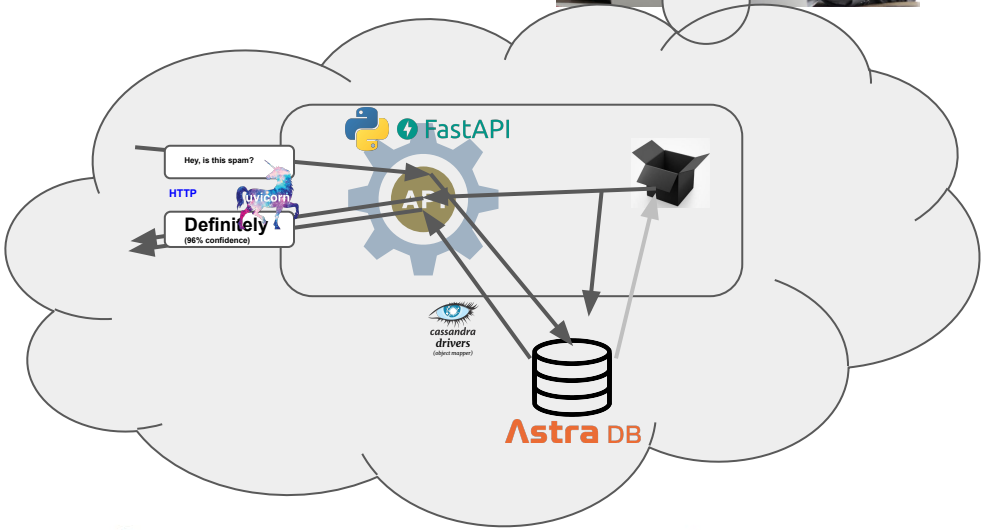
Quiz, Homework, Agenda

API

Bring to production

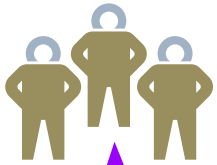
Here, take this

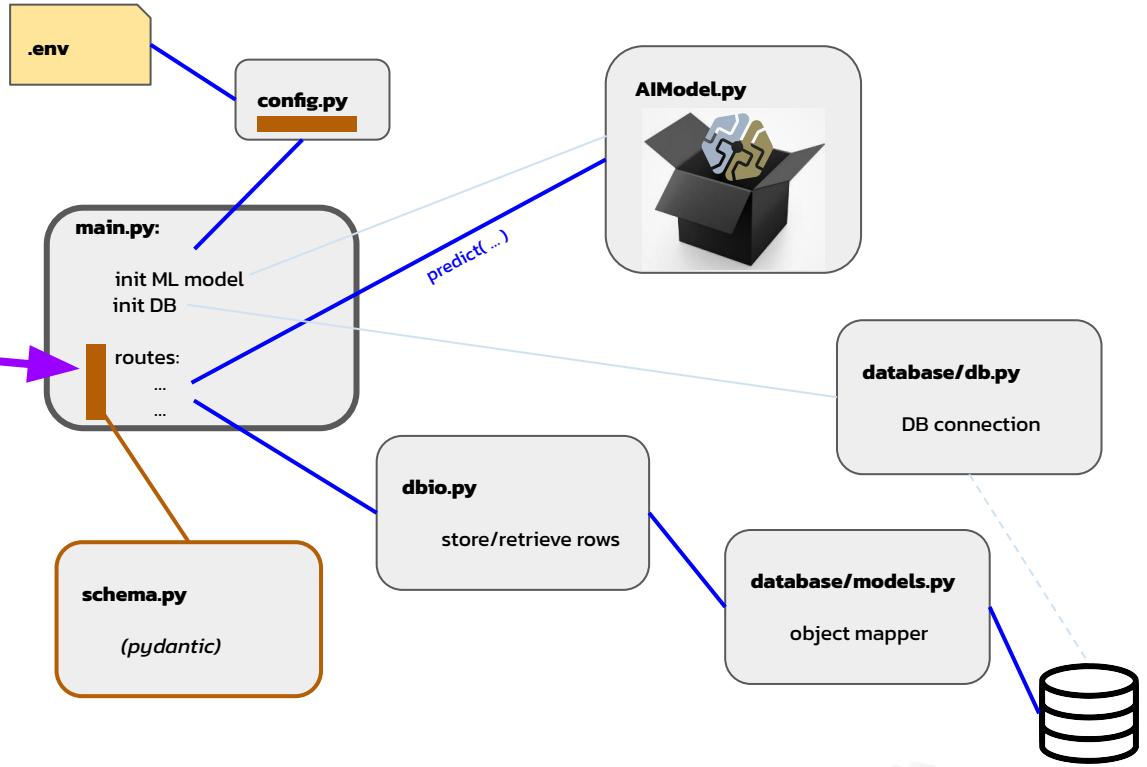




What now?

```
api
├── main.py
├── AIModel.py
├── schema.py
├── dbio.py
├── database
│   ├── db.py
│   └── models.py
├── config.py
├── minimain.py
├── MockSpamAIModel.py
└── tests (...)
```





API structure

FastAPI fundamentals: routes, body validation

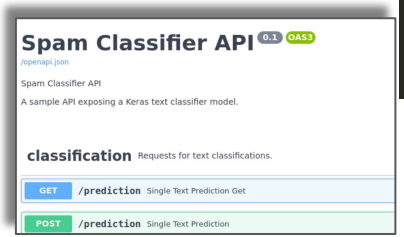
caching on DB to avoid re-processing the same input

API docs, Swagger(Open API) UI

call logs & **`StreamingResponse`**

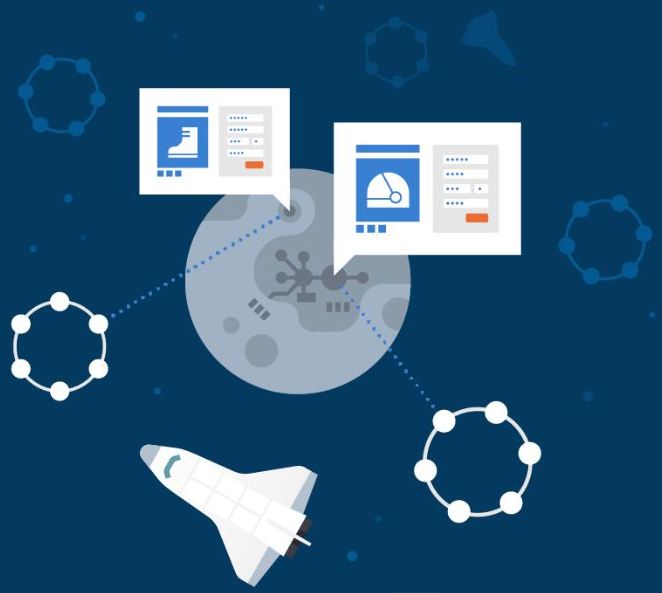
`GET` and query string

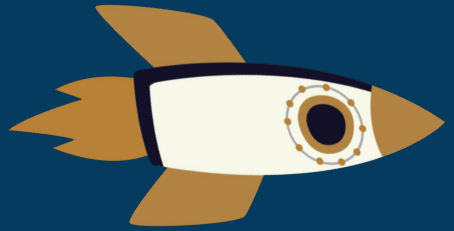
```
settings = getSettings()
cached = None if query.skip_cache else readCachedPrediction(settings.model_versio
storeCallsToLog([query.text], request.client[0])
if not cached:
    result = spamClassifier.predict([query.text], echoInput=query.echo_input)[0]
    cachePrediction(query.text, result)
    result['from_cache'] = False
    return PredictionResult(**result)
else:
    cached['from_cache'] = True
    return PredictionResult(**cached)
```




```
@app.get('/recent_log', response_model=List[CallerLogEntry], tags=['info'])
def get_recent_calls_log(request: Request):
    caller_id = request.client[0]
    called_hour = getThisHour()
    #
    return StreamingResponse(formatCallerLogJSON(caller_id, called_hour))
```

```
@app.get('/prediction', response_model=PredictionResult, tags=['classification'])
def single_text_prediction_get(request: Request, query: SingleTextQuery = Depends()):
    """
```





Hands-on (!github)

API

- ✅ Expose the model with FastAPI
- ✅ Explore Swagger (Open API) UI
- ✅ Look at the database

01

Housekeeping

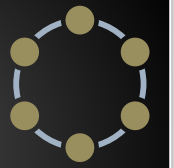
Live and Hands-on

02

Our Goal

Use case and Technologies

03



Database Setup

Data model & Astra DB

04

AI

Train a text classifier

05

API

Bring to production

06

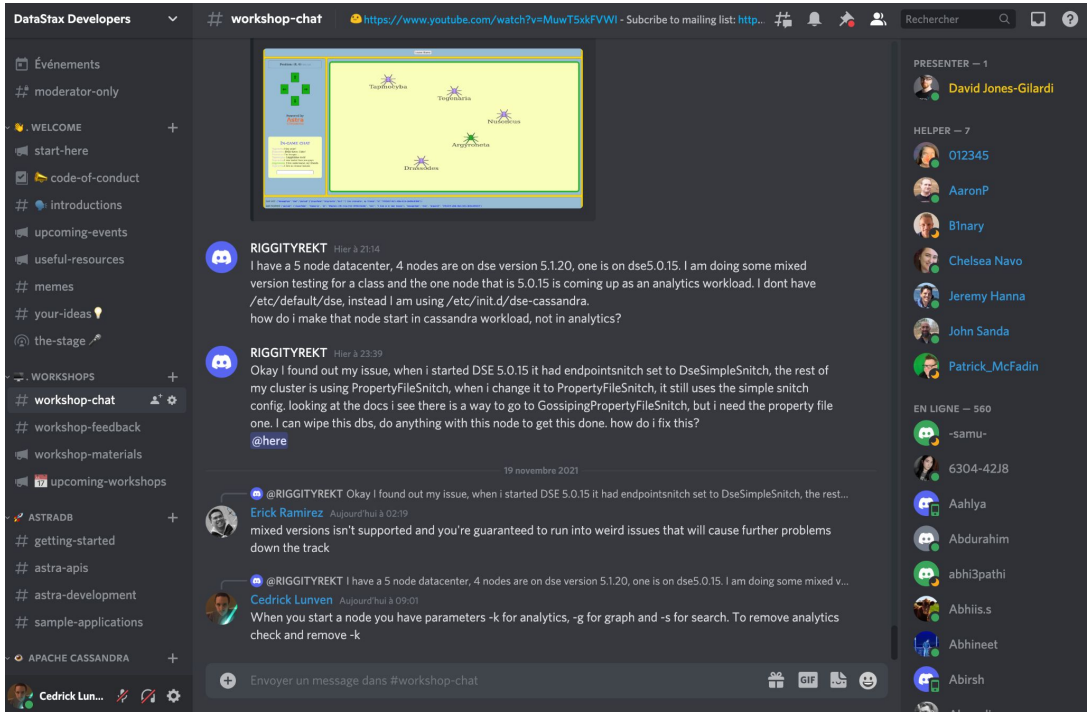
What's next?

Quiz, Homework, Agenda

Agenda

What's next?

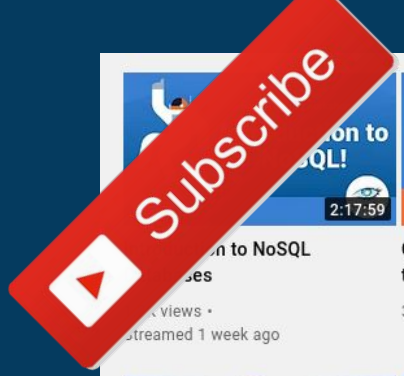
Quiz, Homework, Agenda

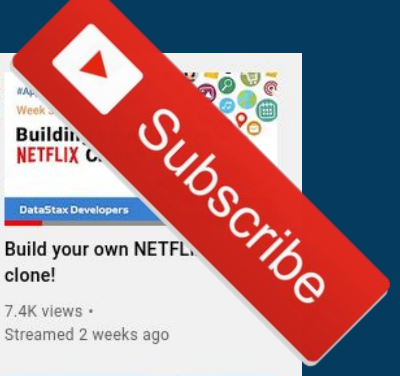


!discord

dtsx.io/discord

Datastax Developers Discord (18k+)





…n to NoSQL …ses

…k views •
…Streamed 1 week ago

Crash Course | Introduction to Cassandra for Developers

331 views • 1 week ago

Introduction to NoSQL Databases

3.5K views •
Streamed 1 week ago

Introduction to NoSQL Databases

10K views •
Streamed 1 week ago

Build your own NETFLIX clone!

4K views •
Streamed 2 weeks ago

Build your own NETFL… clone!

7.4K views •
Streamed 2 weeks ago

Astra Streaming Demo

177 views • 2 weeks ago

Kubernetes Ingress Management with Traefik…

496 views •
Streamed 2 weeks ago

Build your own TikTok clone!

1.9K views •
Streamed 3 weeks ago

Build your own TikTok Clone!

4K views •
Streamed 3 weeks ago

How to use the Connect Driver in Astra DB

113 views • 4 weeks ago

How to use the CQL Console in Astra DB

39 views • 4 weeks ago

How to create an Authentication Token in…

37 views • 4 weeks ago

How to use the Data Loader in Astra DB

62 views • 4 weeks ago

Astra DB Sample App Gallery

36 views • 4 weeks ago

How to use Secure Connect in Astra DB

42 views • 4 weeks ago

Cassandra Day India: CL Room (Workshops)

2.4K views •
Streamed 4 weeks ago

Cassandra Day India: RF Room (Talks)

1.3K views •
Streamed 1 month ago


INTRODUCTION TO
cassandra
WORKSHOP
DataStax Developers

cassandra
Introduction to
NoSQL
WORKSHOP
DataStax Developers

cassandra
DATA MODELING
WORKSHOP
DataStax Developers

cassandra
SQL to NoSQL
WORKSHOP
DataStax Developers

K8SSANDRA
WORKSHOP
DataStax Developers

cassandra
Spring Data +
Cassandra
WORKSHOP
DataStax Developers

cassandra
Quarkus +
Cassandra
WORKSHOP
DataStax Developers

Vue.JS
WORKSHOP
DataStax Developers

ReactNative
WORKSHOP
DataStax Developers

GraphQL
WORKSHOP
DataStax Developers

Building a
ToDO App
WORKSHOP
DataStax Developers

TikTok Clone
WORKSHOP
DataStax Developers

Netflix Clone
WORKSHOP
DataStax Developers

Event Streaming
Learning Series
DataStax Developers

Event
Streaming
WORKSHOP
DataStax Developers

Reactive
Spring
WORKSHOP
DataStax Developers

AI as an
API
WORKSHOP
DataStax Developers


Badges

Thank You!