Validation of Machine Learning services

Do not let your Machine Learning service going into production without being tested

---

José Antonio Perdiguero López

http://www.perdy.io

https://github.com/perdy

https://www.linkedin.com/in/perdy

@ perdy@perdy.io

May 20, 2019

Software Architect @ Piksel

Index

Introduction

Introduction

Do we really know what **Artificial Intelligence** is?

Introduction

Do we really know what **Artificial Intelligence** is?

And, specifically, do we know what **Machine Learning** is?

Introduction

Do we really know what **Artificial Intelligence** is?

And, specifically, do we know what **Machine Learning** is?

How can we **verify** and **validate** services whose response depends on a Machine Learning model?

Goals

1. **Discover new tools** for building REST APIs and design Tests.

Goals

1. **Discover new tools** for building REST APIs and design Tests.
2. **Understand** what is Artificial Intelligence and **Machine Learning**.

Goals

1. **Discover new tools** for building REST APIs and design Tests.
2. **Understand** what is Artificial Intelligence and **Machine Learning**.
3. **Build a service** that relies on a Machine Learning model.

Goals

1. **Discover new tools** for building REST APIs and design Tests.
2. **Understand** what is Artificial Intelligence and **Machine Learning**.
3. **Build a service** that relies on a Machine Learning model.
4. **Generate some tests** that verify and validate the service and the model.

Our Tools

The glue

Our Tools

The glue

The shield

Our Tools

The glue

The power

The shield

Our Tools

The glue

The power

The shield

The mind

Machine Learning

What is Artificial Intelligence?

Artificial intelligence is the simulation of human intelligence processes by computer systems. These processes include **learning** (the acquisition of information and rules for using the information), **reasoning** (using rules to reach approximate or definite conclusions) and **self-correction**.

What is Artificial Intelligence?

Artificial intelligence is the simulation of human intelligence processes by computer systems. These processes include **learning** (the acquisition of information and rules for using the information), **reasoning** (using rules to reach approximate or definite conclusions) and **self-correction**.

AI can be categorized as either **weak** or **strong**.

What is Artificial Intelligence?

Artificial intelligence is the simulation of human intelligence processes by computer systems. These processes include **learning** (the acquisition of information and rules for using the information), **reasoning** (using rules to reach approximate or definite conclusions) and **self-correction**.

AI can be categorized as either **weak** or **strong**.

Strong AI
Also known as artificial general intelligence. Is an AI system with generalized human cognitive abilities. When presented with an unfamiliar task, a strong AI system is able to find a solution without human intervention.

What is Artificial Intelligence?

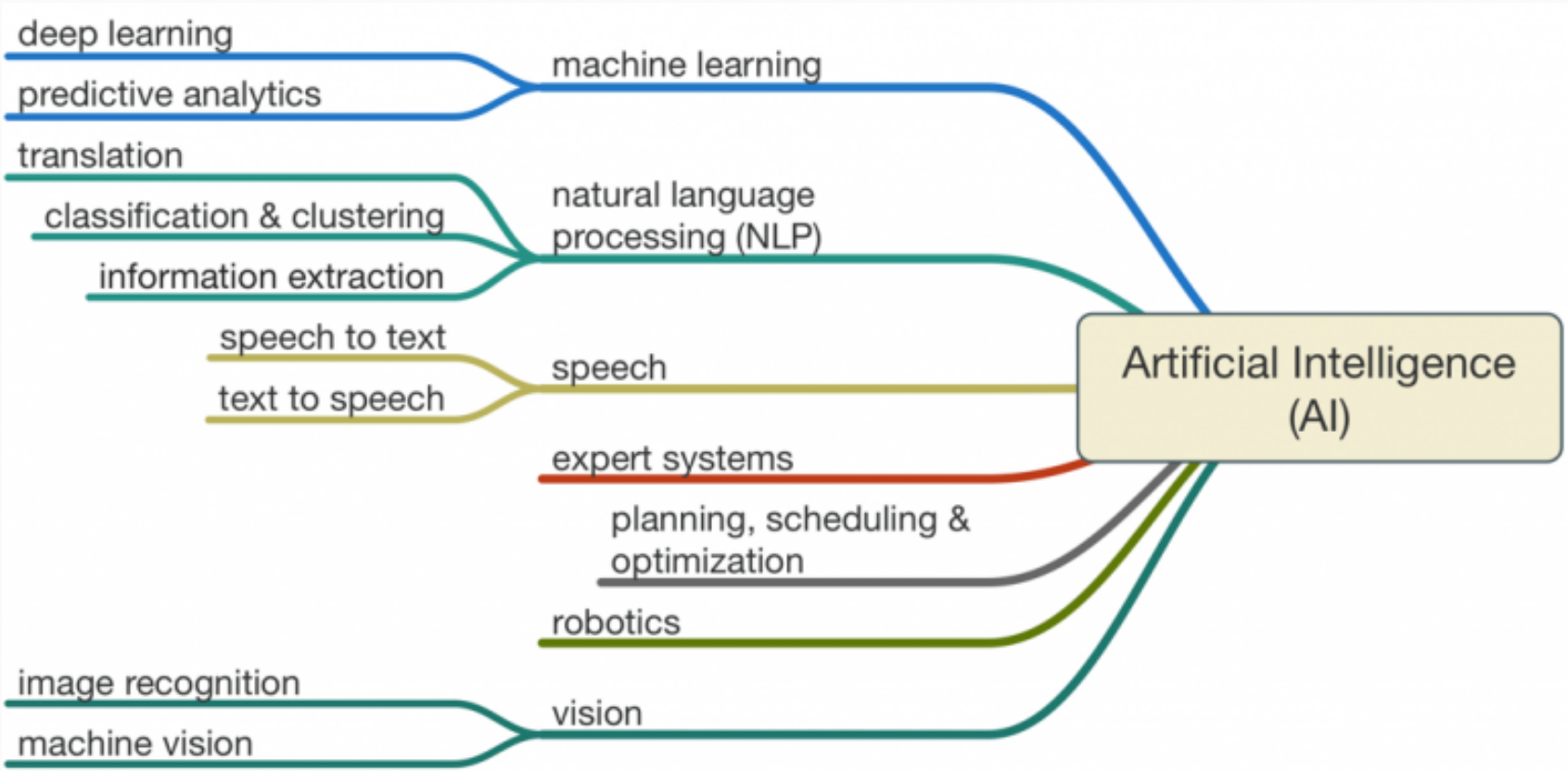
Artificial intelligence is the simulation of human intelligence processes by computer systems. These processes include **learning** (the acquisition of information and rules for using the information), **reasoning** (using rules to reach approximate or definite conclusions) and **self-correction**.

AI can be categorized as either **weak** or **strong**.

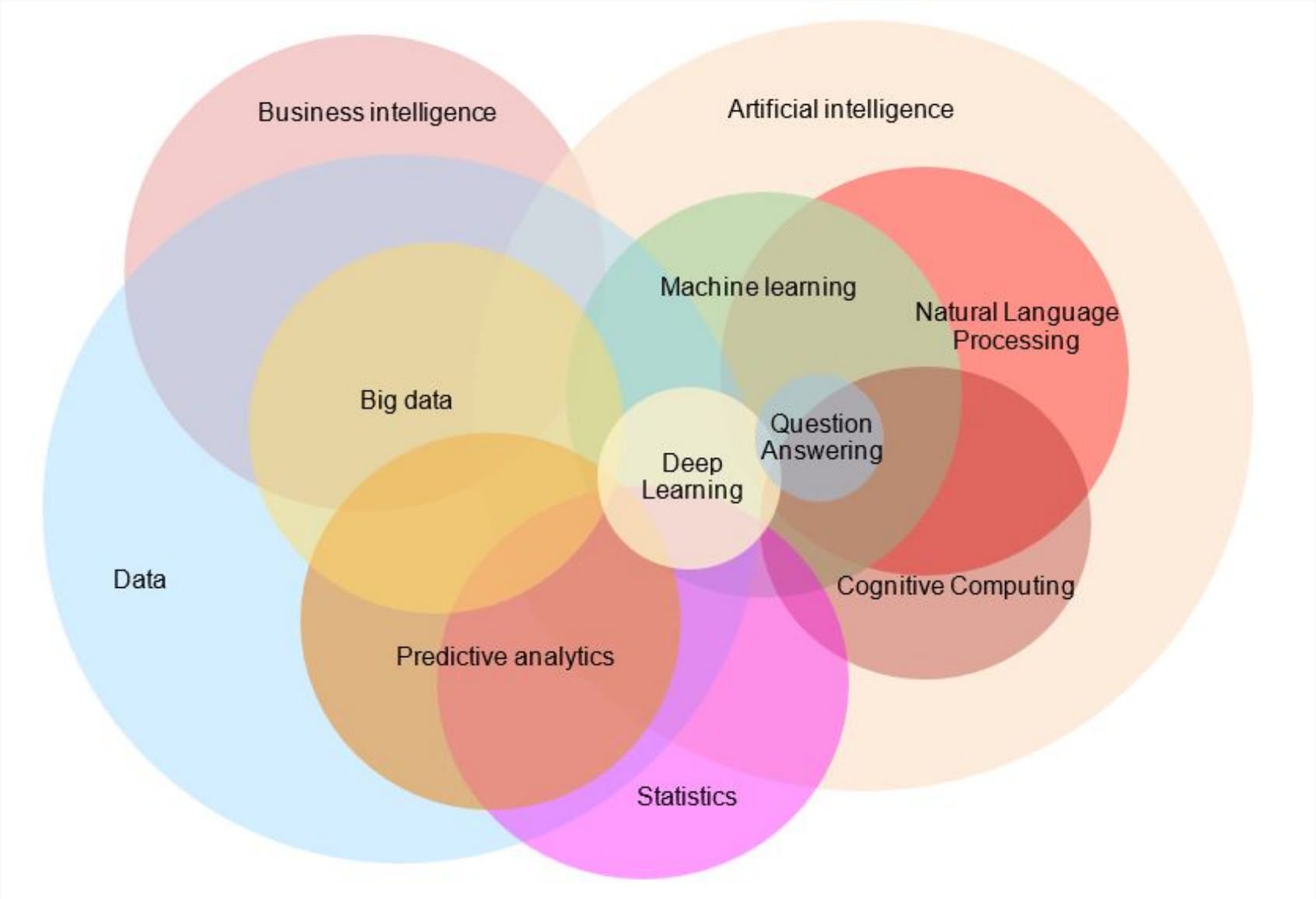
Strong AI
Also known as artificial general intelligence. Is an AI system with generalized human cognitive abilities. When presented with an unfamiliar task, a strong AI system is able to find a solution without human intervention.

Weak AI
Also known as narrow AI. Is an AI system that is designed and trained for a particular task.

AI in perspective



AI in perspective



AI in perspective

Artificial Intelligence

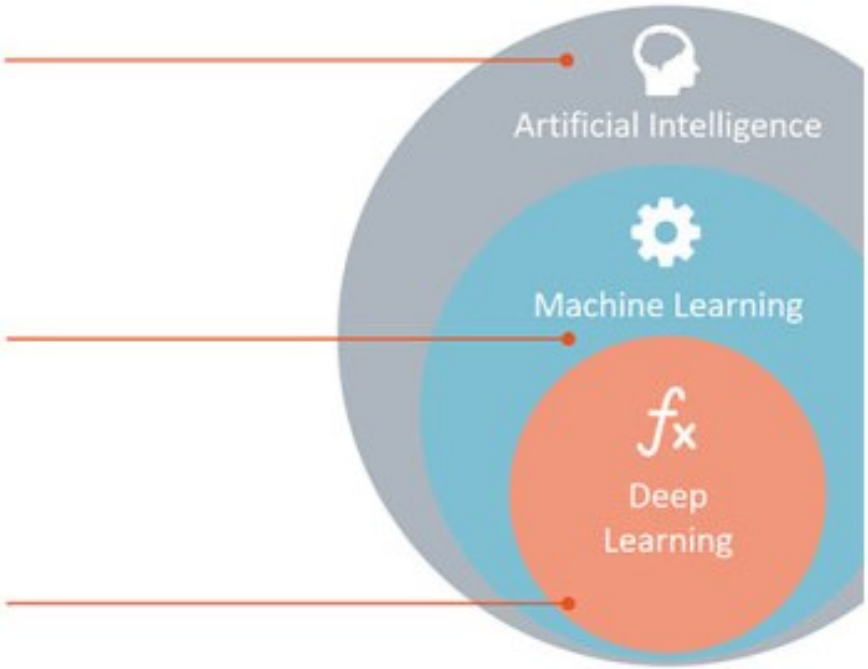
Any technique which enables computers to mimic human behavior.

Machine Learning

Subset of AI techniques which use statistical methods to enable machines to improve with experiences.

Deep Learning

Subset of ML which make the computation of multi-layer neural networks feasible.



What is Machine Learning?

The science of getting a computer to act without programming. There are three types of machine learning algorithms:

What is Machine Learning?

The science of getting a computer to act without programming. There are three types of machine learning algorithms:

Supervised learning
Data sets are labeled so that patterns can be detected and used to label new data sets.

What is Machine Learning?

The science of getting a computer to act without programming. There are three types of machine learning algorithms:

Supervised learning
Data sets are labeled so that patterns can be detected and used to label new data sets.

Unsupervised learning
Data sets aren't labeled and are sorted according to similarities or differences.

What is Machine Learning?

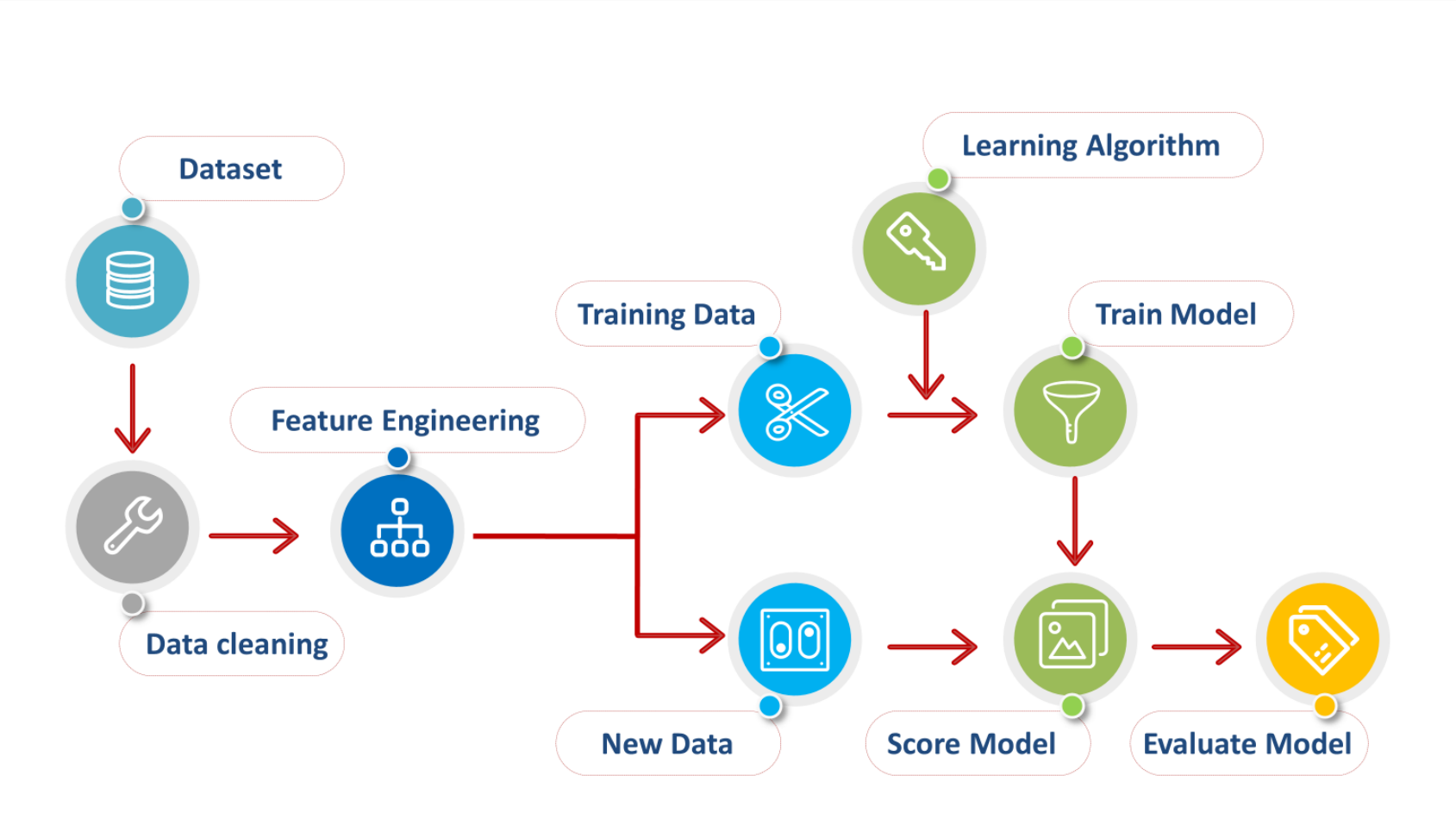
The science of getting a computer to act without programming. There are three types of machine learning algorithms:

Supervised learning
Data sets are labeled so that patterns can be detected and used to label new data sets.

Unsupervised learning
Data sets aren't labeled and are sorted according to similarities or differences.

Reinforcement learning
Data sets aren't labeled but, after performing an action or several actions, the AI system is given feedback.

Supervised Learning

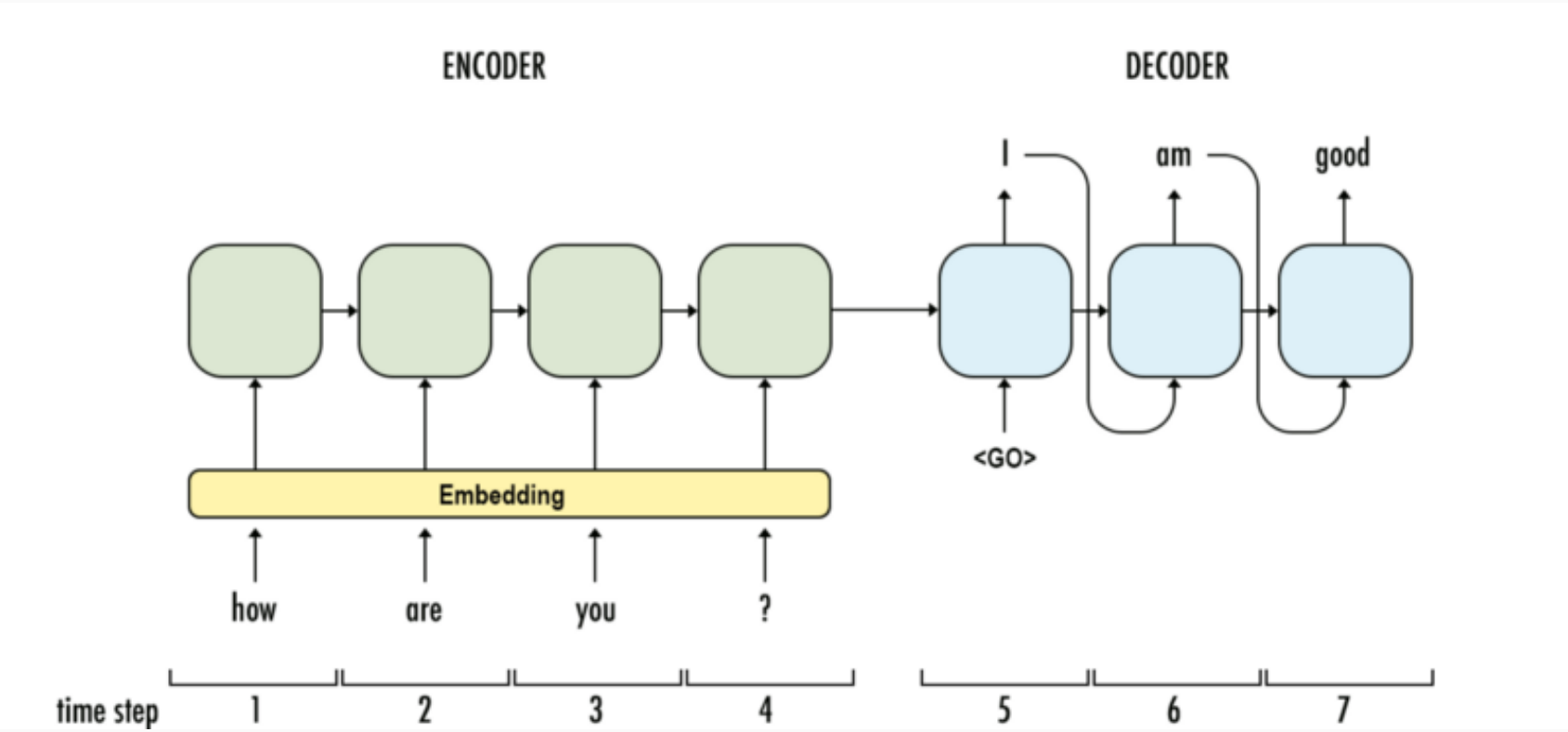


The Model

We are going to build a model that performs a sentiment analysis over a text and concludes if it is positive or negative.

Input: A text (a list of integers representing each word).

Output: 0 (negative) or 1 (positive).



Training Dataset

The dataset used for training this model is based on movie's reviews from IMDB.

Text input	Input	Output
the as you with out themselves...	[1, 14, 22, 16, 43, 530, ...]	1
the thought solid thought sena...	[1, 194, 1153, 194, 8255, ...]	0
the as there in at by br of su...	[1, 14, 47, 8, 30, 31, 7, ...]	0
the of bernadette mon they hal...	[1, 4, 18609, 16085, 33, ...]	1
the sure themes br only acting...	[1, 249, 1323, 7, 61, 113, ...]	0

Building a Machine Learning Service

How to expose the ML model?

Machine Learning models can be used either as an internal piece of a service or as a service itself.

How to expose the ML model?

Machine Learning models can be used either as an internal piece of a service or as a service itself.

If it is used as an **internal piece** you won't notice it, such as scoring or recommendation systems within bigger products like Spotify or Netflix.

How to expose the ML model?

Machine Learning models can be used either as an internal piece of a service or as a service itself.

If it is used as an **internal piece** you won't notice it, such as scoring or recommendation systems within bigger products like Spotify or Netflix.

But you can also find them as a **service that exposes an API** to directly interact with the model. There are many examples of that in AWS, Google Cloud, Azure...

Wrapping up a ML model

One of the most widely adopted way of serving a ML model is to wrap it into a **REST API** with specific methods for calling the model.

The Service

Our service will expose a single endpoint that let us interact with the model.

Request

Verb	`GET`
URL	`https://service.url/analyze/`
Params	`text=The%20girl%20is%20having%20fun%20while%20playing`

Response

```
{
  "text": "The girl is having fun while playing",
  "sentiment": "Positive",
  "score": 0.6321590542793274
}
```

Building a REST API with Flama

To build a REST API we need to define:

1. A **component** that loads our ML model.

Building a REST API with Flama

To build a REST API we need to define:

1. A **component** that loads our ML model.
2. The **data schema** for our response.

Building a REST API with Flama

To build a REST API we need to define:

1. A **component** that loads our ML model.
2. The **data schema** for our response.
3. The **view** function that will be called through requests to `/analyze/` endpoint.

Building a REST API with Flama

To build a REST API we need to define:

1. A **component** that loads our ML model.
2. The **data schema** for our response.
3. The **view** function that will be called through requests to `/analyze/` endpoint.
4. The whole API **application**.

Everything put together is less than 100 lines of python code.

ML Component

```
class SentimentAnalysisModel:
    def __init__(self, model, words: typing.Dict[str, int]):
        self.model = model
        self.words = words

    def predict(self, text: str) -> typing.Tuple[float, str]:
        x = text.lower().split()
        x = [self.words.get(i, 0)
             if self.words.get(i, 0) <= VOCABULARY_LENGTH else 0 for i in x]
        x = sequence.pad_sequences([x], maxlen=MAX_WORDS)
        score = self.model.predict(x)
        sentiment = "Positive" if self.model.predict_classes(x)[0][0] == 1 \
            else "Negative"

        return score, sentiment

class SentimentAnalysisModelComponent(Component):
    def __init__(self, model_path: str, words_path: str):
        self.model = load_model(model_path)
        self.model._make_predict_function()
        with open(words_path) as f:
            self.words = json.load(f)

    def resolve(self) -> SentimentAnalysisModel:
        return SentimentAnalysisModel(model=self.model, words=self.words)
```

Data Schema

```
class SentimentAnalysis(Schema):
    text = fields.String(
        title="text",
        description="Text to analyze"
    )
    score = fields.Float(
        title="score",
        description="Sentiment score in range [0,1]"
    )
    sentiment = fields.String(
        title="sentiment",
        description="Sentiment class (Positive or Negative)"
    )
```

Analysis View

```
def analyze(text: str, model: SentimentAnalysisModel) -> SentimentAnalysis:
    """
    tags:
        - sentiment-analysis
    summary:
        Sentiment analysis.
    description:
        Performs a sentiment analysis on a given text.
    responses:
        200:
            description: Analysis result.
    """
    text = unquote(text)
    score, sentiment = model.predict(text)

    return {
        "text": text,
        "score": score,
        "sentiment": sentiment,
    }
```

API Application

```
app = Flama(
    components=[SentimentAnalysisModelComponent("model.h5", "words.json")],
    title="Sentiment Analysis",
    version="0.1",
    description="A sentiment analysis API for movies reviews",
)

app.add_route("/analyze/", analyze, methods=["GET"])
```

Testing the Service

Testing Considerations

The most commons development cases of Machine Learning services are those where the building of the model and the service are done completely separated and even by different teams.

That implies we aren't in control of the training process so that we cannot test the model until both are merged.

Validation vs Verification

Criteria	Verification	Validation
Definition	The process of evaluating products of a development phase to determine whether they meet the specified requirements.	The process of evaluating software during or at the end of the development process to determine whether it satisfies specified business requirements.
Objective	To ensure that the product is being built according to the requirements and design specifications.	To demonstrate that the product fulfills its intended use when placed in its intended environment.
Question	Are we building the product right?	Are we building the right product?

Test Specification: Verification

```
## Endpoint Verification
Tags: functional, verification

Verify if the endpoint that allows interaction with Sentiment Analyzer
is properly defined based on specifications. It must provide a query
parameter **text** that acts as the input of the model and it cannot be
empty. The response must be a JSON containing three attributes:
**text**, **score** and **sentiment**.

* Request sentiment analysis with text ”Perdy is testing this” returns ”200”
* Response schema contains attributes
    |Attribute|
    |---------|
    |text     |
    |score    |
    |sentiment|
* Request sentiment analysis with text ”” returns ”400”
```

Test Specification: Validation

```
## Model Validation
Tags: ml, validation

Validate the model predictions against a set of fixed data. This data set
must contains a minimum list of well-known pairs of input and output to
check that after retraining the model it will continue behaving the same
way against these inputs.

* Analyze and validate the following texts <table:data/sentiment_analysis.csv>
```

Step Implementation

```
@step("Response schema contains attributes <table>")
def assert_response_schema(table):
    response = data_store.scenario["response"]

    for attribute in table.get_column_values_with_name("Attribute"):
        assert attribute in response
```

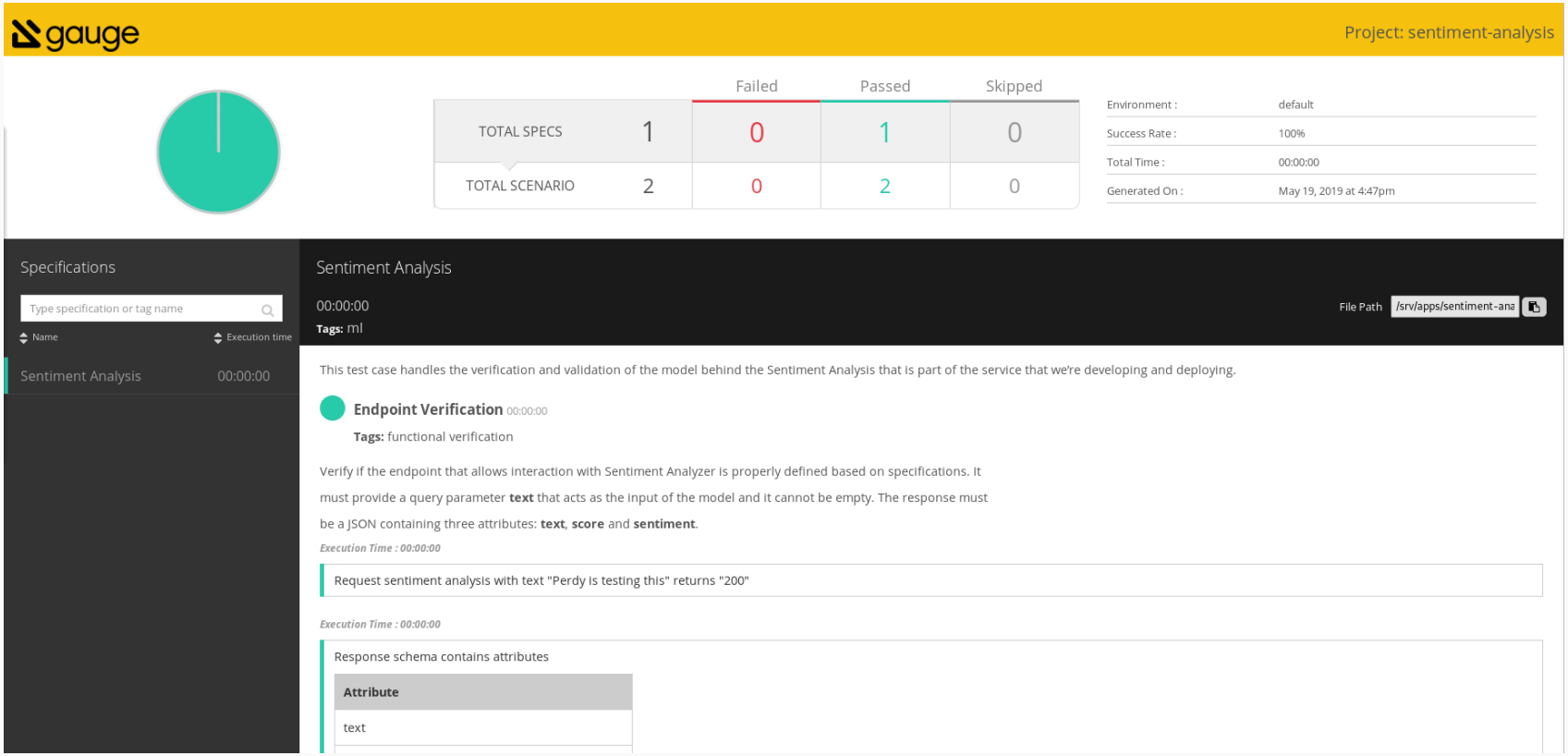
Console Output

```
•100% → ./make test
# Sentiment Analysis
  ## Endpoint Verification       ✔ ✔ ✔
  ## Model Validation    ✔

Successfully generated html-report to ⇒ /srv/apps/sentiment-analysis/reports/html-report/index.html
Specifications: 1 executed      1 passed        0 failed        0 skipped
Scenarios:      2 executed      2 passed        0 failed        0 skipped

Total time taken: 505ms
```

HTML Output


gauge
Project: sentiment-analysis
Failed
Passed
Skipped
TOTAL SPECS 1 0 1 0
TOTAL SCENARIO 2 0 2 0
Environment : default
Success Rate : 100%
Total Time : 00:00:00
Generated On : May 19, 2019 at 4:47pm
Specifications
Type specification or tag name
Name
Execution time
Sentiment Analysis 00:00:00
Sentiment Analysis
00:00:00
Tags: ml
File Path /srv/apps/sentiment-ana
This test case handles the verification and validation of the model behind the Sentiment Analysis that is part of the service that we're developing and deploying.
Endpoint Verification 00:00:00
Tags: functional verification
Verify if the endpoint that allows interaction with Sentiment Analyzer is properly defined based on specifications. It must provide a query parameter text that acts as the input of the model and it cannot be empty. The response must be a JSON containing three attributes: text, score and sentiment.
Execution Time : 00:00:00
Request sentiment analysis with text "Perdy is testing this" returns "200"
Execution Time : 00:00:00
Response schema contains attributes
Attribute
text

Support open source projects:

★ **Give a star**

📣 **Spread the word**

https://flama.perdy.io/

https://getgauge.io/

https://www.tensorflow.org

https://www.python.org/