
Organizado por
nexo qa
training & events
TEST
ACADEMY
MADRID - 26 NOVIEMBRE 2019
El día para los Testers
#TestAcademy
www.testacademy.es
Patrocinado por
sogeti
Part of Capgemini

Machine Learning Services Validation

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

Support open source projects by giving a star ⭐ and spreading the word 📣

https://flama.perdy.io/

https://getgauge.io/

https://www.tensorflow.org

https://www.python.org/

Index

Introduction

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.

1. **Understand** what is Artificial Intelligence and Machine Learning.

1. **Build a Machine Learning model** for a solving a complex problem.

1. **Develop a service** that relies on a Machine Learning model.

1. **Generate some tests** that verify and validate the service and the model.

Tools

The Glue

The Mind

The Power

The Shield

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**.

AI can be categorized as either weak or strong.

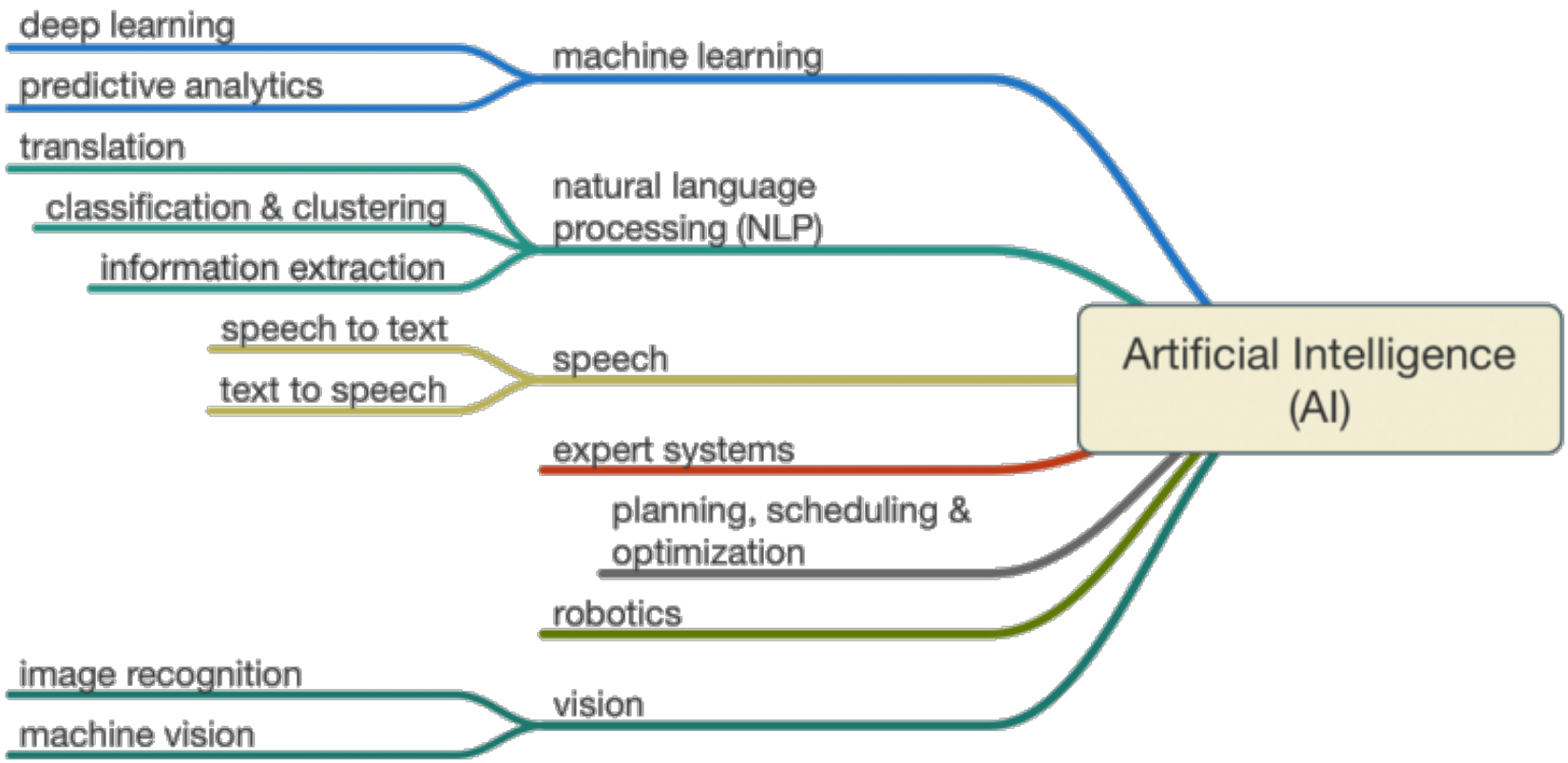
Strong AI

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.

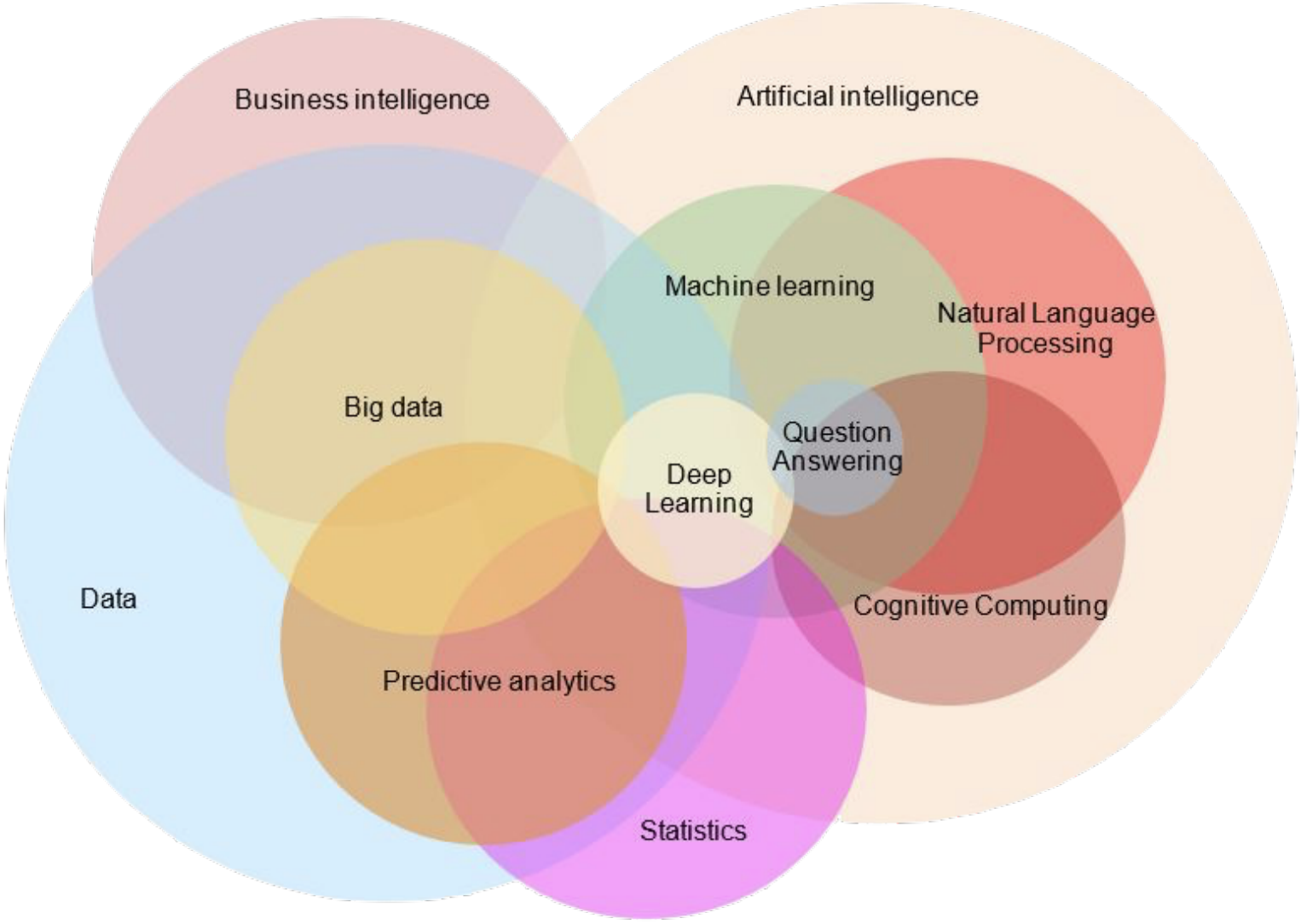
Weak AI

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

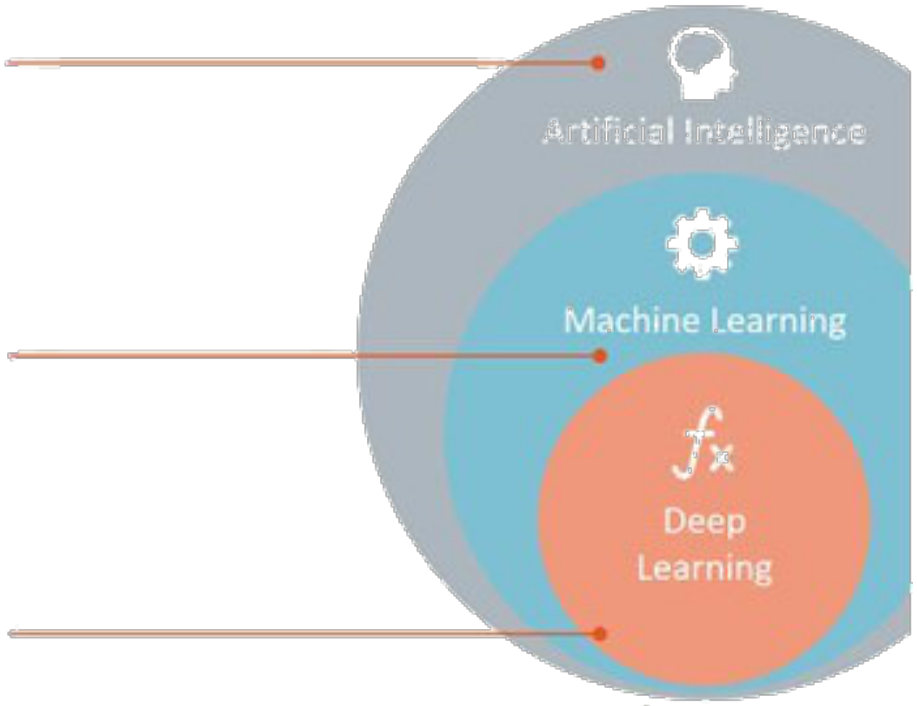
Any technique which enables computers to mimic human behavior.

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:

Supervised learning

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

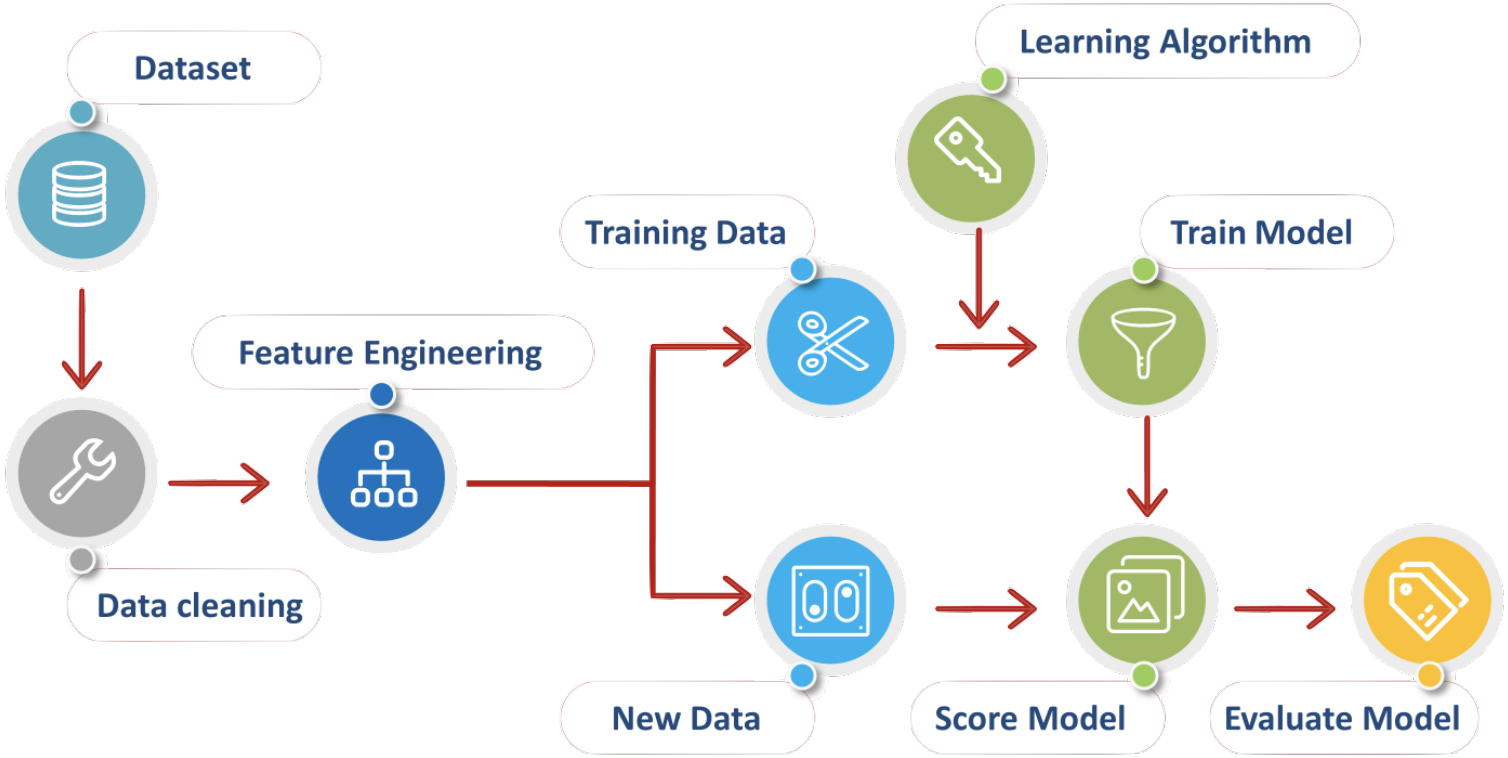
Unsupervised learning

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

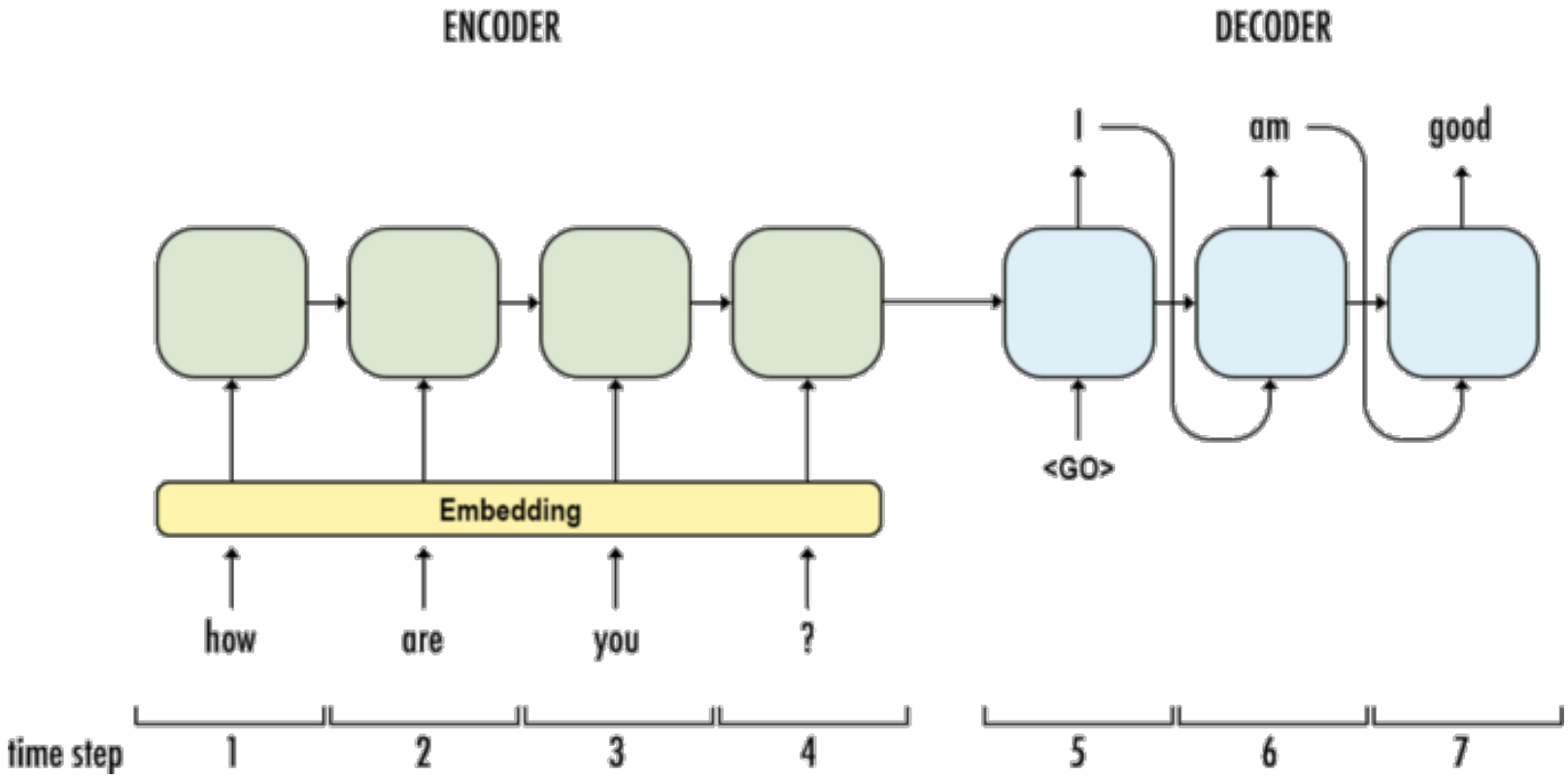


The Model

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).



Building A Machine Learning Model

Training Dataset

The dataset used for training this model is based on movie's reviews from IMDB and it will have the following shape:

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 The Training Dataset

A first step in a dataset building process could be to define how large it will be.

In [ ]:
```
VOCABULARY_LENGTH = 20000
```

Keras provides some datasets so it's possible to skip the data gathering process.

In [ ]:
```
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words = VOCABULARY_LENGTH)
```

As the dataset used for training will consist of integers that represents a word, it could be interesting to have mappings of **Word -> ID** and **ID -> Word**.

In [ ]:
```
word2id = imdb.get_word_index()
id2word = {i: word for word, i in word2id.items()}
```

All input documents must have the same length so it's necessary to limit the maximum review length by truncating longer reviews and padding shorter reviews with a null value (0).

In [ ]:

```
MAX_WORDS = 500
X_train = sequence.pad_sequences(X_train, maxlen=MAX_WORDS)
X_test = sequence.pad_sequences(X_test, maxlen=MAX_WORDS)
```

Design Model

Input: Sequence of words (integer ids) whose length are MAX_WORDS.

Output: Binary label (0 means *Negative* and 1 means *Positive*)

In [ ]:
```
model=Sequential()
model.add(Embedding(VOCABULARY_LENGTH, 32, input_length=MAX_WORDS))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

Compile And Train Our Model

We first need to compile our model by specifying the loss function and optimizer we want to use while training, as well as any evaluation metrics we'd like to measure.

In [ ]:
```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Once compiled, we can run the training process.

In [ ]:
```
BATCH_SIZE = 64
NUM_EPOCHS = 5

validation = X_train[:BATCH_SIZE], y_train[:BATCH_SIZE]
training = X_train[BATCH_SIZE:], y_train[BATCH_SIZE:]

model.fit(*training, validation_data=validation, batch_size=BATCH_SIZE, epochs=NUM
_EPOCHS);
```

Test Model

Once the training process is finished the model should be tested using unseen data to measure its accuracy.

In [ ]:
```
scores = model.evaluate(X_test, y_test, verbose=0)
f"Test accuracy: {scores[1]:.5f}"
```

Prediction Example

In [ ]:
```
score, sentiment = predict(model, word2id, "A girl is happy while playing with her toys")

f"The result is '{sentiment}' with a score of {score:.2f}"
```

Save Model And Words Mapping

In order to use the model within a service it's necessary to export it and save the model and the words dictionary in files.

In [ ]:
```
model.save("sentiment_analysis_model.h5")

with open("sentiment_analysis_words.json", "w") as f:
    json.dump(word2id, f, indent=4)
```

Developing The 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.

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.

1. The **data schema** for our response.

1. The **view** function that will be called through requests to /analyze/ endpoint.

1. The whole API **application**.

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

ML Component

In [ ]:
```
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 els
e 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 "N
egative"
        return score, sentiment
```

In [ ]:
```
class SentimentAnalysisModelComponent(Component):
    def __init__(self, model_path: str, words_path: str):
        self._model_path = model_path
        with open(words_path) as f:
            self.words = json.load(f)

    @property
    def model(self):
        if not hasattr(self, "_model"):
            from keras.models import load_model
            self._model = load_model(self._model_path)
            self._model._make_predict_function()
        return self._model

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

Data Schema

In [ ]:
```
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

In [ ]:

```
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

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

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

Run The Service

In [ ]:
```
Process(target=uvicorn.run, kwargs={"app": app, "host": "0.0.0.0", "port": 8000},
daemon=True).start()
```

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 re- quirements.	The process of evaluating software during or at the end of the development process to determine whether it sat- isfies specified business re- quirements.
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

```
markdown
## 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

```
markdown
## 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
```

Run Tests

In [ ]:
```
!gauge run tests/specs
```

HTML Report (tests/reports/html-report/index.html)


Organizado por
nexo qa
training & events
TEST
ACADEMY
MADRID - 26 NOVIEMBRE 2019 El día para los Testers
¡Gracias!
#TestAcademy
www.testacademy.es
Patrocinado por
sogeti
Part of Capgemini

Machine Learning Services Validation

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

Support open source projects by giving a star ⭐ and spreading the word 📣

https://flama.perdy.io/

https://getgauge.io/

https://www.tensorflow.org

https://www.python.org/