Z

Webinar

Desenvolvimento

Criando um Bot para interagir com o z/OS

22.07.2020

10:00 - 11:30

GMT-3

Ecossistema para IBM Z & LinuxONE

SOBRE NÓS

Nossa missão é ampliar e cultivar o ecossistema de IBM **Z** disseminando conteúdo técnico, divulgando oportunidades, realizando eventos e promovendo o networking entre profissionais experientes e a nova geração de Mainframers.

EMPRESAS
Apoiar organizações no processo de identificação e formação de novos profissionais em Mainframe.

ACADEMIA
Apoiar instituições de ensino a inserirem IBM **Z** na grade curricular, com o intuito de formar uma nova geração de Mainframers.

COMUNIDADE
Promover encontros virtuais e presenciais para troca de experiência e conhecimento entre profissionais e estudantes.

INICIATIVA Z

Desenvolvendo carreiras em TI Enterprise

Ecossistema para IBM Z & LinuxONE

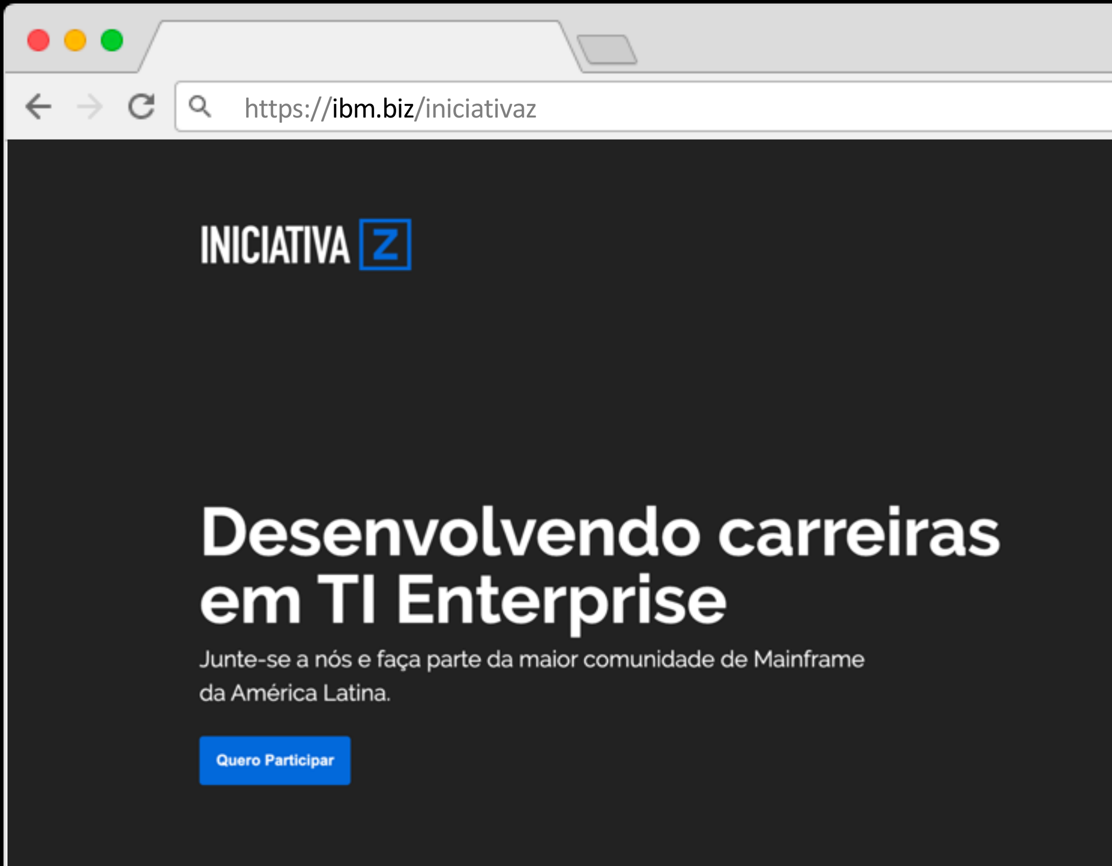
NOSSO TIME

Junte-se a nós e seja parte da maior comunidade de Mainframe da América Latina.

Bill Pereira
Iniciativa Z – Developer Advocate
in /in/bill-pereira/

Ludmila Salimena
Iniciativa Z – Client Advocate
in /in/ludmilasalimena/

Rafael Ireno
Iniciativa Z – Academic Advocate
in /in/rafaelireno/



https://ibm.biz/iniciativaz

INICIATIVA Z

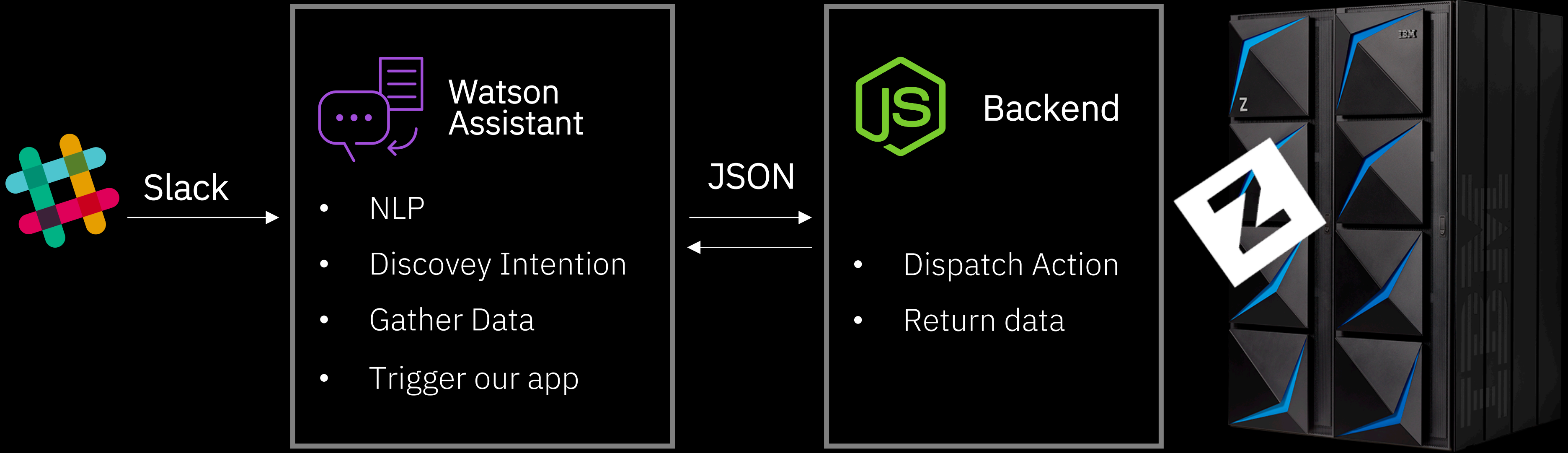
Desenvolvendo carreiras em TI Enterprise

Junte-se a nós e faça parte da maior comunidade de Mainframe da América Latina.

Agenda

Our ChatBot
NLP and Watson
The Backend
Zowe SDK
Connecting to slack
Some real cases…

Our ChatBot



- NLP
- Discovey Intention
- Gather Data
- Trigger our app

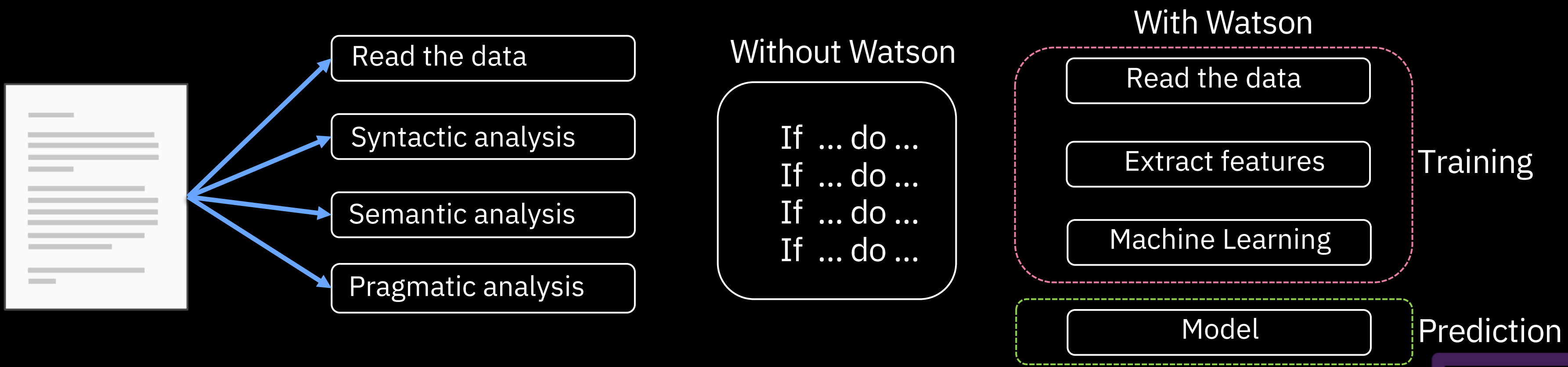
- Dispatch Action
- Return data

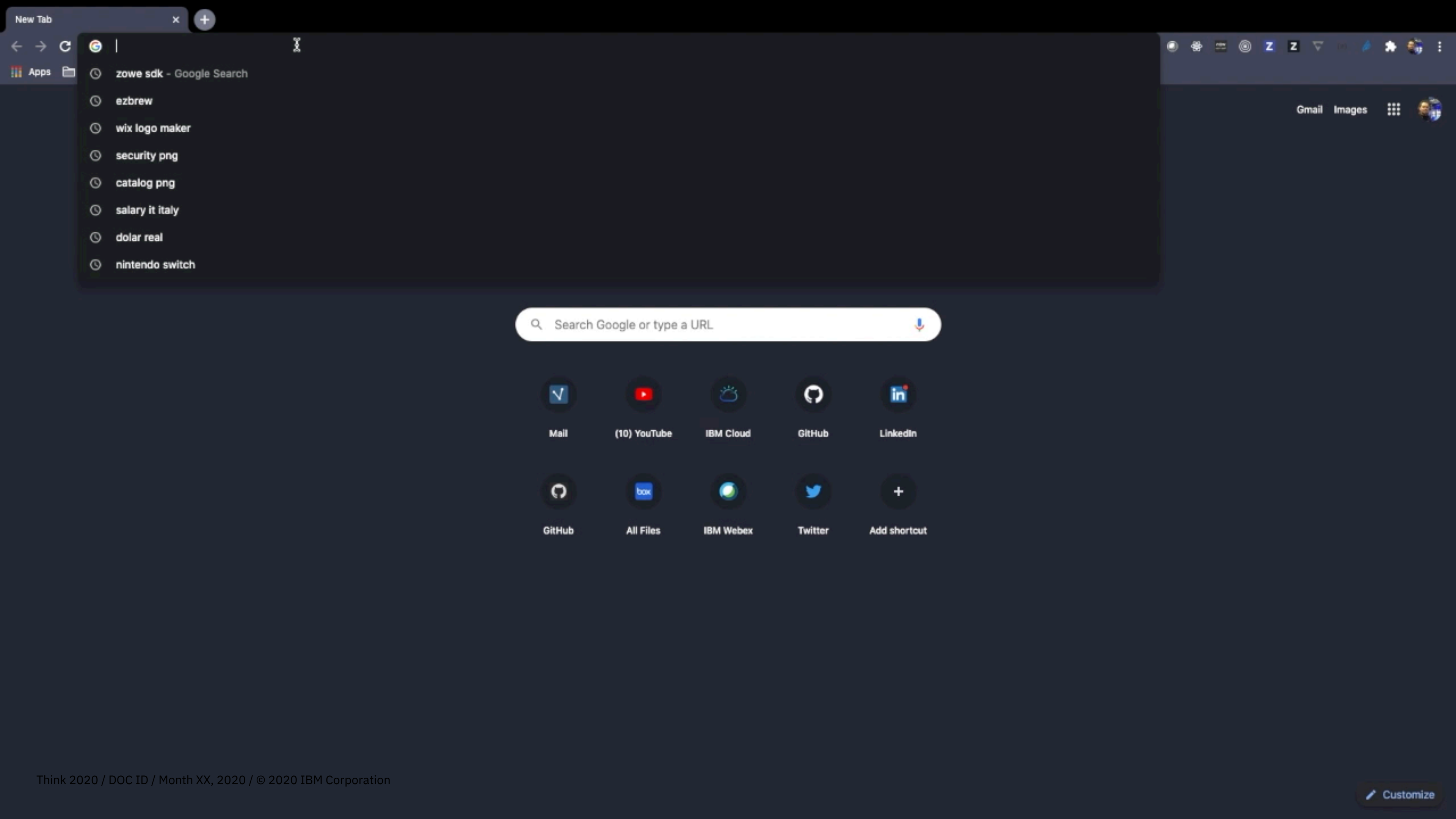
NLP and Watson

NLP and Watson

- Analyze semantic features of text input, including categories, concepts, emotion, entities, keywords, metadata, relations, semantic roles, and sentiment.
- Support a variety of languages depending on which features are being analyzed, including English, Arabic, Chinese (simplified), Dutch, French, German, Italian, Japanese, Korean, Portuguese, Russian, Spanish, and Swedish - with more to come.

Challenges of NLP





zowe sdk - Google Search
ezbrew
wix logo maker
security png
catalog png
salary it italy
dolar real
nintendo switch

The Backend

node JS

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

express

@zowe/cli

axios

dotenv

POST /zowe

MVS_COMMAND

READ_DATASET

USS_COMMAND

OPEN **MAINFRAME** PROJECT

Zowe

Zowe SDK

Basic Setup

```
const zowe = require("@zowe/cli");
```

```
const profileSSH = {
  host: process.env.MFHOST,
  user: process.env.MFUSER,
  password: process.env.MFPWD,
  rejectUnauthorized: false,
};

const SSHsession = zowe.SshSession.createBasicSshSession(profileSSH);

const profile = {
...profileSSH,
port: process.env.MFPORT,
};

const zOSMFsession = zowe.ZosmfSession.createBasicZosmfSession(profile);
```

Basic Setup

```
const zowe = require("@zowe/cli");

const profileSSH = {
  host: process.env.MFHOST,
  user: process.env.MFUSER,
  password: process.env.MFPWD,
  rejectUnauthorized: false,
};

const SSHsession = zowe.SshSession.createBasicSshSession(profileSSH);

const profile = {
...profileSSH,
port: process.env.MFPORT,
};

const zOSMFsession = zowe.ZosmfSession.createBasicZosmfSession(profile);
```

Basic Setup

```
const zowe = require("@zowe/cli");

const profileSSH = {
  host: process.env.MFHOST,
  user: process.env.MFUSER,
  password: process.env.MFPWD,
  rejectUnauthorized: false,
};

const SSHsession = zowe.SshSession.createBasicSshSession(profileSSH);

const profile = {
...profileSSH,
port: process.env.MFPORT,
};

const zOSMFsession = zowe.ZosmfSession.createBasicZosmfSession(profile);
```

Basic Setup

```
const zowe = require("@zowe/cli");

const profileSSH = {
  host: process.env.MFHOST,
  user: process.env.MFUSER,
  password: process.env.MFPWD,
  rejectUnauthorized: false,
};

const SSHsession = zowe.SshSession.createBasicSshSession(profileSSH);

const profile = {
...profileSSH,
port: process.env.MFPORT,
};

const zOSMFsession = zowe.ZosmfSession.createBasicZosmfSession(profile);
```

Basic Setup

```
const zowe = require("@zowe/cli");

const profileSSH = {
  host: process.env.MFHOST,
  user: process.env.MFUSER,
  password: process.env.MFPWD,
  rejectUnauthorized: false,
};

const SSHsession = zowe.SshSession.createBasicSshSession(profileSSH);

const profile = {
...profileSSH,
port: process.env.MFPORT,
};
```

```
const zOSMFsession = zowe.ZosmfSession.createBasicZosmfSession(profile);
```

Dispatching the actions

```
dispatch: async (req, res) => {

const { intention, command, dsname } = req.body;

var data;
 switch (intention) {

  case "MVS_COMMAND":
   data = await issueMVSCommand(command);
   break;

  case "READ_DATASET":
   data = await readDataset(dsname);
   break;

  case "USS_COMMAND":
   data = await issueUSSCommand(command);
   break;

  default:
   data = `Invalid options: ${req.body}`;
   break;
  }

 return res.json(data);
}
```

Dispatching the actions

```
dispatch: async (req, res) => {
const { intention, command, dsname } = req.body;

var data;
 switch (intention) {

  case "MVS_COMMAND":
   data = await issueMVSCommand(command);
   break;

  case "READ_DATASET":
   data = await readDataset(dsname);
   break;

  case "USS_COMMAND":
   data = await issueUSSCommand(command);
   break;

  default:
   data = `Invalid options: ${req.body}`;
   break;
  }

 return res.json(data);
}
```

Dispatching the actions

```
dispatch: async (req, res) => {

const { intention, command, dsname } = req.body;

var data;
 switch (intention) {

   case "MVS_COMMAND":
    data = await issueMVSCommand(command);
    break;

   case "READ_DATASET":
    data = await readDataset(dsname);
    break;

   case "USS_COMMAND":
    data = await issueUSSCommand(command);
    break;

   default:
    data = `Invalid options: ${req.body}`;
    break;
   }

 return res.json(data);
}
```

Dispatching the actions

```
dispatch: async (req, res) => {

const { intention, command, dsname } = req.body;

var data;
 switch (intention) {

  case "MVS_COMMAND":
   data = await issueMVSCommand(command);
   break;

  case "READ_DATASET":
   data = await readDataset(dsname);
   break;

  case "USS_COMMAND":
   data = await issueUSSCommand(command);
   break;

  default:
   data = `Invalid options: ${req.body}`;
   break;
  }

 return res.json(data);
}
```

Dispatching the actions

```
dispatch: async (req, res) => {

const { intention, command, dsname } = req.body;

var data;
 switch (intention) {

  case "MVS_COMMAND":
   data = await issueMVSCommand(command);
   break;

  case "READ_DATASET":
   data = await readDataset(dsname);
   break;

  case "USS_COMMAND":
   data = await issueUSSCommand(command);
   break;

  default:
   data = `Invalid options: ${req.body}`;
   break;
  }

 return res.json(data);
}
```

```
const issueMVSCommand = async (command) => {
const { commandResponse } = await zowe.IssueCommand.issueAndCollect(
    zOSMFsession,
    { command },
    {}
  );
  return commandResponse;
};
```

Dispatching the actions

```
dispatch: async (req, res) => {

const { intention, command, dsname } = req.body;

var data;
 switch (intention) {

  case "MVS_COMMAND":
   data = await issueMVSCommand(command);
   break;

  case "READ_DATASET":
   data = await readDataset(dsname);
   break;

  case "USS_COMMAND":
   data = await issueUSSCommand(command);
   break;

  default:
   data = `Invalid options: ${req.body}`;
   break;
  }

 return res.json(data);
}
```

Dispatching the actions

```
dispatch: async (req, res) => {

const { intention, command, dsname } = req.body;

var data;
 switch (intention) {

   case "MVS_COMMAND":
    data = await issueMVSCommand(command);
    break;

   case "READ_DATA":
    data = await readDataset(dsname);
    break;

   case "USS_COMMAND":
    data = await issueUSSCommand(command);
    break;

   default:
    data = `Invalid options: ${req.body}`;
    break;
   }

 return res.json(data);
}
```

```
const readDataset = async (dsname) => {
  const data = await zowe.Get.dataSet(zOSMFsession, dsname);
  return data.toString();
};
```

Dispatching the actions

```
dispatch: async (req, res) => {

const { intention, command, dsname } = req.body;

var data;
 switch (intention) {

  case "MVS_COMMAND":
   data = await issueMVSCommand(command);
   break;

  case "READ_DATASET":
   data = await readDataset(dsname);
   break;

  case "USS_COMMAND":
   data = await issueUSSCommand(command);
   break;

  default:
   data = `Invalid options: ${req.body}`;
   break;
  }

 return res.json(data);
}
```

Dispatching the actions

```
dispatch: async (req, res) => {

const { intention, command, dsname } = req.body;

var data;
 switch (intention) {

  case "MVS_COMMAND":
   data = await issueMVSCommand(command);
   break;

  case "READ_DATASET":
   data = await readDataset(dsname);
   break;

  case "USS_COMMAND":
   data = await issueUSSCommand(command);
   break;

  default:
   data = `Invalid options: ${req.body}`;
   break;
  }

 return res.json(data);
}
```

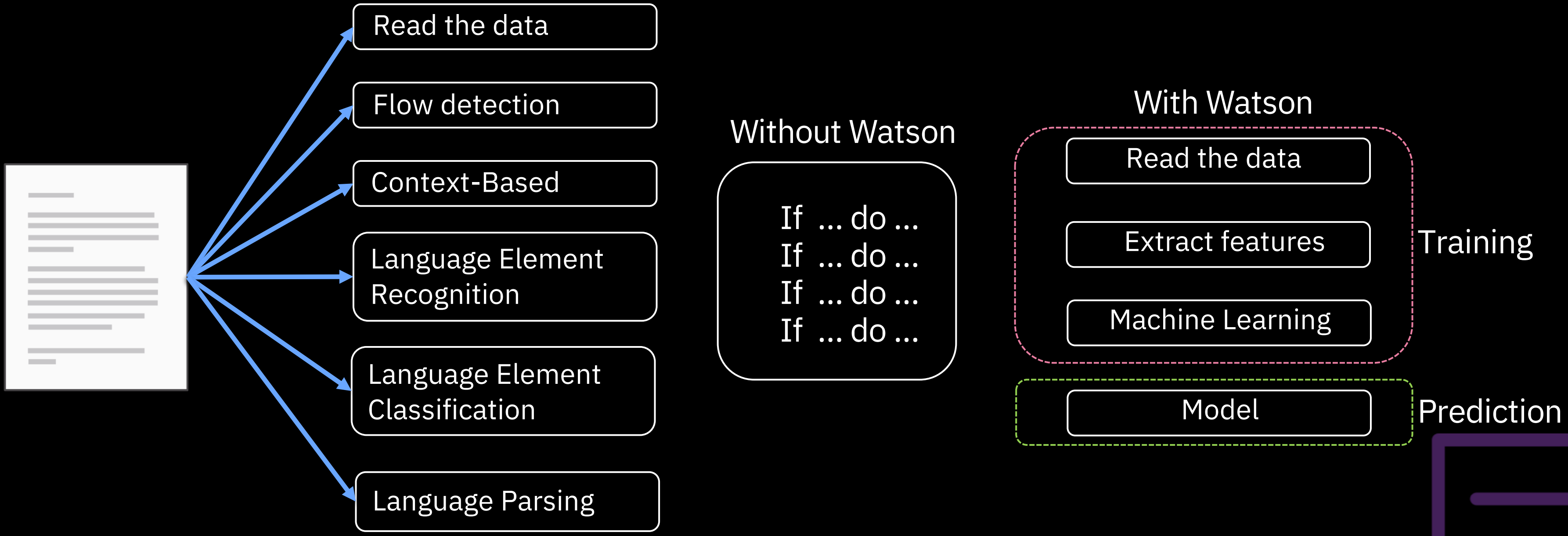
```
const issueUSSCommand = async (command) => {
  const response = [];
  await zowe.Shell.executeSsh(SSHsession, command, (data) => {
    response.push(data);
  });
  return response;
};
```

Connecting to SLACK

DÚVIDAS?

IBM®

Challenges of NLP


Read the data
Flow detection
Context-Based
Language Element Recognition
Language Element Classification
Language Parsing
Without Watson
If ... do ...
If ... do ...
If ... do ...
If ... do ...
With Watson
Read the data
Extract features
Machine Learning
Training
Model
Prediction