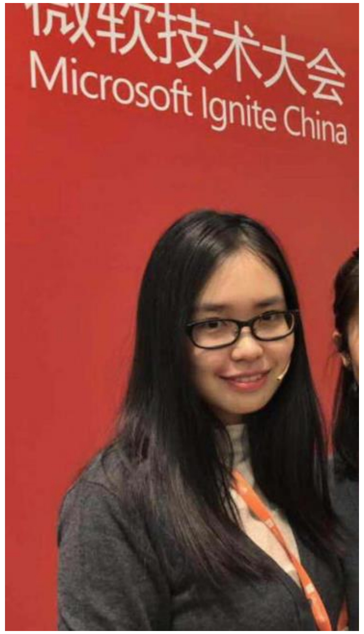
Use Intel Analytics Zoo to build an intelligent QA Bot for Microsoft Azure

Jan 10th, 2019

About Us

Kai Huang

Software Engineer from Intel Data Analytics Technology Team



Yuqing Wei

Software Engineer from Microsoft C+AI Team

Outline

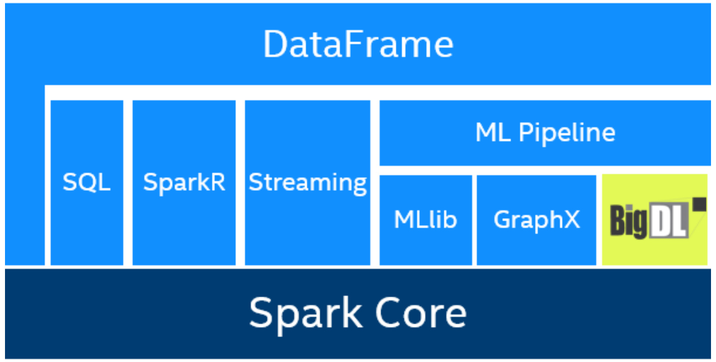
- Introduction to Analytics Zoo.
- How to develop NLP modules using Analytics Zoo.
- Engineering experience in building a chat bot.
- Bot Demo.

BigDL

Bringing Deep Learning To Big Data Platform

- Distributed deep learning framework for Apache Spark.
- Make deep learning more accessible to big data users and data scientists
    - Write deep learning applications as standard Spark programs.
    - Run on existing Spark/Hadoop clusters (no changes needed).
- High performance (on CPU)
    - Powered by Intel MKL and multi-threaded programming.
- Efficient scale-out
    - Leveraging Spark for distributed training & inference.

https://github.com/intel-analytics/BigDL
https://bigdl-project.github.io/



Analytics Zoo

A unified analytics + AI platform for

distributed TensorFlow, Keras and BigDL on Apache Spark

Feature Engineering	Feature transformations for • Image, 3D images, text, time series, speech, etc.
High-Level Pipeline APIs	• Keras, autograd and transfer learning APIs for model definition. • Native deep learning support in Spark DataFrames and ML Pipelines. • POJO style API for model serving/inference pipelines.
Built-In Deep Learning Models	Image classification, object detection, text classification, recommendations, text matching, anomaly detection, seq2seq etc.
Backends	Seamlessly unites Spark, TensorFlow, Keras and BigDL programs into an integrated pipeline.

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

Feature Engineering

1. **Read images as *`ImageSet`***

```
from zoo.common.nncontext import init_nncontext
from zoo.feature.image import *

sc = init_nncontext()
local_image_set = ImageSet.read(image_path)
distributed_image_set = ImageSet.read(image_path, sc, 2)
```

2. **Built-in *`ImageProcessing`* operations**

```
transformer = ChainedPreprocessing([ImageBytesToMat(),
                   ImageColorJitter(),
                   ImageExpand(max_expand_ratio=2.0),
                   ImageResize(300, 300, -1),
                   ImageHFlip()])
new_local_image_set = transformer(local_image_set)
new_distributed_image_set = transformer(distributed_image_set)
```

Image Augmentations Using Built-in Image Transformations (w/ OpenCV on Spark)

ANALYTICS ZOO

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

Keras-Style API

Use Keras-Style API to create an Analytics Zoo model and train, evaluate or tune it in a distributed fashion.

```
from zoo.pipeline.api.keras.models import Sequential
from zoo.pipeline.api.keras.layers import *

model = Sequential()
model.add(Reshape((1, 28, 28), input_shape=(28, 28, 1)))
model.add(Convolution2D(6, 5, 5, activation="tanh", name="conv1_5x5"))
model.add(MaxPooling2D())
model.add(Convolution2D(12, 5, 5, activation="tanh", name="conv2_5x5"))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(100, activation="tanh", name="fc1"))
model.add(Dense(10, activation="softmax", name="fc2"))

model.compile(optimizer, loss, metrics)
model.fit(x, batch, epoch)
model.predict(x, batch)
model.evaluate(x, batch)
```

ANALYTICS ZOO

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

Autograd API

Autograd API provides automatic differentiation for math operations to easily define custom layers or losses.

```
import zoo.pipeline.api.autograd as A

log = A.log(in_node + 1.0)
dot = A.batch_dot(embed1, embed2, axes=[2, 2])
```

```
from zoo.pipeline.api.autograd import *

def mean_absolute_error(y_true, y_pred):
    result = mean(abs(y_true - y_pred), axis=1)
    return result
```

ANALYTICS ZOO

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

Transfer Learning API

Use transfer learning APIs to easily customize pretrained models for *feature extraction or fine-tuning*:

```
from zoo.pipeline.api.net import *
from zoo.pipeline.api.keras.layers import Dense, Input, Flatten
from zoo.pipeline.api.keras.models import Model

# Load a pretrained model
full_model = Net.load_caffe(def_path, model_path)

# Remove the last few layers
model = full_model.new_graph(outputs=["pool5"]).to_keras())

# Freeze the first few layers
model.freeze_up_to(["res4f"])

# Append a few layers
input = Input(shape=(3, 224, 224))
resnet = model.to_keras()(input)
flatten = Flatten()(resnet)
logits = Dense(2)(flatten)
new_model = Model(input, logits)
```

ANALYTICS ZOO

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

Distributed TensorFlow

Running TensorFlow model on Spark for distributed training and inference.

1. **Data wrangling and analysis using PySpark**

```
from zoo.common.nncontext import init_nncontext
from zoo.pipeline.api.net import TFDataset

sc = init_nncontext()

# Each record in the train_rdd consists of a list of NumPy ndrrays
train_rdd = sc.parallelize(file_list)\
 .map(lambda x: read_image_and_label(x))
 .map(lambda image_label: decode_to_ndarrays(image_label))

# TFDataset represents a distributed set of elements,
# in which each element contains one or more TensorFlow Tensor objects.
dataset = TFDataset.from_rdd(train_rdd,
                             names=["features", "labels"],
                             shapes=[[28, 28, 1], [1]],
                             types=[tf.float32, tf.int32],
                             batch_size=BATCH_SIZE)
```

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

Distributed TensorFlow

Running TensorFlow model on Spark in a distributed fashion.

2. **Deep learning model development using TensorFlow**

```
import tensorflow as tf

slim = tf.contrib.slim

images, labels = dataset.tensors
labels = tf.squeeze(labels)
with slim.arg_scope(lenet.lenet_arg_scope()):
    logits, end_points = lenet.lenet(images, num_classes=10, is_training=True)

loss = tf.reduce_mean(tf.losses.sparse_softmax_cross_entropy(logits=logits,
                                                              labels=labels))
```

3. **Distributed training on Spark and BigDL**

```
from zoo.pipeline.api.net import TFOptimizer
from bigdl.optim.optimizer import MaxIteration, Adam, MaxEpoch, TrainSummary

optimizer = TFOptimizer(loss, Adam(1e-3))
optimizer.set_train_summary(TrainSummary("/tmp/az_lenet", "lenet"))
optimizer.optimize(end_trigger=MaxEpoch(5))
saver = tf.train.Saver()
saver.save(optimizer.sess, "/tmp/lenet/")
```

ANALYTICS ZOO

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

Distributed TensorFlow

Running TensorFlow model on Spark in a distributed fashion.

4. **Distributed inference**

```
dataset = TFDataset.from_rdd(testing_rdd,
                             names=["features"],
                             shapes=[[28, 28, 1]],
                             types=[tf.float32])

predictor = TFPredictor.from_outputs(sess, [logits])
predictions_rdd = predictor.predict()
```

For Keras users:

```
optimizer = TFOptimizer.from_keras(keras_model, dataset)

predictor = TFPredictor.from_keras(model, dataset)
predictions_rdd = predictor.predict()
```

ANALYTICS ZOO

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

NNFrames API

Native DL support in Spark DataFrames and ML Pipelines

1. **Load images into *DataFrames* using *NNImageReader***

```
from zoo.common.nncontext import *
from zoo.pipeline.nnframes import *

sc = init_nncontext()
imageDF = NNImageReader.readImages(image_path, sc)
```

2. **Process loaded data using *DataFrame* transformations**

```
getName = udf(lambda row: ...)
df = imageDF.withColumn("name", getName(col("image")))
```

3. **Processing image using built-in *feature engineering* operations**

```
from zoo.feature.image import *

transformer = ChainedPreprocessing(
        [RowToImageFeature(), ImageChannelNormalize(123.0, 117.0, 104.0),
         ImageMatToTensor(), ImageFeatureToTensor()])
```

ANALYTICS ZOO

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

NNFrames API

Native DL support in Spark DataFrames and ML Pipelines

4. **Define model using *Keras-style API***

```
from zoo.pipeline.api.keras.layers import *
from zoo.pipeline.api.keras.models import Sequential

model = Sequential()
model.add(Convolution2D(32, 3, 3, activation='relu', input_shape=(1, 28, 28)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten()).add(Dense(10, activation='softmax')))
```

5. **Train model using *Spark ML Pipelines***

```
Estimater = NNEstimater(model, CrossEntropyCriterion(), transformer) \
                .setLearningRate(0.003).setBatchSize(40).setMaxEpoch(1) \
                .setFeaturesCol("image")
nnModel = estimater.fit(df)
```

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

Models Interoperability Support

- Load existing TensorFlow, Keras, Caffe, Torch, ONNX model
  - Useful for inference and model fine-tuning.
  - Allows for transition from single-node for distributed application deployment.
  - Allows for model sharing between data scientists and production engineers.

```
from zoo.pipeline.api.net import Net

Net.load_tf(path, inputs=None, outputs=None,
            byte_order="little_endian", bin_file=None)

Net.load_keras(hdf5_path, json_path=None, by_name=False)

Net.load_caffe(def_path, model_path)

Net.load_torch(path)
```

ANALYTICS ZOO

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

Built-in Deep Learning Models

- **Object detection API**
  - High-level API and pretrained models (e.g., SSD, Faster-RCNN, etc.) for object detection.
- **Image classification API**
  - High-level API and pretrained models (e.g., VGG, Inception, ResNet, MobileNet, etc.) for image classification.
- **Recommendation API**
  - High-level API and pre-defined models (e.g., Neural Collaborative Filtering, Wide and Deep Learning, etc.) for recommendation.
- **Text classification API**
  - High-level API and pre-defined models (using CNN, LSTM, etc.) for text classification.

ANALYTICS ZOO

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

Object Detection API

1. **Load pretrained model in *Detection Model Zoo***

```
from zoo.common.nncontext import init_nncontext
from zoo.models.image.objectdetection import *

sc = init_nncontext()
model = ObjectDetector.load_model(model_path)
```

2. **Off-the-shell inference using the loaded model**

```
image_set = ImageSet.read(img_path, sc)
output = model.predict_image_set(image_set)
```

3. **Visualize detection results**

```
config = model.get_config()
visualizer = Visualizer(config.label_map(), encoding="jpg")
visualized = visualizer(output).get_image(to_chw=False).collect()
```

ANALYTICS ZOO

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

Reference Use Cases

- ***Anomaly Detection***
  - Using LSTM network to detect anomalies in time series data.
- ***Fraud Detection***
  - Using feed-forward neural network to detect frauds in credit card transaction data.
- ***Recommendation***
  - Use Analytics Zoo Recommendation API (i.e., Neural Collaborative Filtering, Wide and Deep) for recommendations on data with explicit feedback.
- ***Sentiment Analysis***
  - Sentiment analysis on movie reviews using neural network models (e.g. CNN, LSTM, GRU, Bi-LSTM).
- ***Variational AutoEncoder***
  - Use VAE to generate digital numbers and faces.

https://github.com/intel-analytics/analytics-zoo/tree/master/apps

ANALYTICS ZOO

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

Public Cloud Deployment

Deployed on AliCloud* E-MapReduce*

https://yq.aliyun.com/articles/73347

Listed in Microsoft* Azure* Marketplace*

https://azure.microsoft.com/en-us/blog/bigdl-spark-deep-learning-library-vm-now-available-on-microsoft-azure-marketplace/

Available on Google* Cloud Dataproc*

https://cloud.google.com/blog/big-data/2018/04/using-bigdl-for-deep-learning-with-apache-spark-and-google-cloud-dataproc

Optimized for Amazon* EC2* C5 instanced, and listed in AWS* Marketplace*

https://aws.amazon.com/blogs/machine-learning/leveraging-low-precision-and-quantization-for-deep-learning-using-the-amazon-ec2-c5-instance-and-bigdl/

Deployed on IBM* Data Science Expetience*

https://medium.com/ibm-data-science-experience/using-bigdl-in-data-science-experience-for-deep-learning-on-spark-f1cf30ad6ca0

Available on Telefonica* Open Cloud*

https://support.telefonicaopencloud.com/en-us/ecs/doc/download/20180329/20180329111611_166372a698.pdf

ANALYTICS ZOO

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

Customer Use Cases

Industrial Inspection Platform in Midea* and KUKA*
https://software.intel.com/en-us/articles/industrial-inspection-platform-in-midea-and-kuka-using-distributed-tensorflow-on-analytics

Object Detection and Image Feature Extraction in JD
https://software.intel.com/en-us/articles/building-large-scale-image-feature-extraction-with-bigdl-at-jdcom

Image Similarity Based House Recommendation for MLSlistings
https://software.intel.com/en-us/articles/using-bigdl-to-build-image-similarity-based-house-recommendations

3D Medical Image Analysis in UCSF
https://conferences.oreilly.com/strata/strata-ca/public/schedule/detail/64023

ANALYTICS ZOO

https://github.com/intel-analytics/analytics-zoo
https://analytics-zoo.github.io/

Background

- Chat Bot is often used for recent intelligent customer platforms.
- To enhance user experience and relieve human workload.
- To provide technical support for Azure users effectively and efficiently.
- AI modules provided by Analytics Zoo: text classification, question answering, intent extraction, named entity recognition, etc.

Why neural networks?

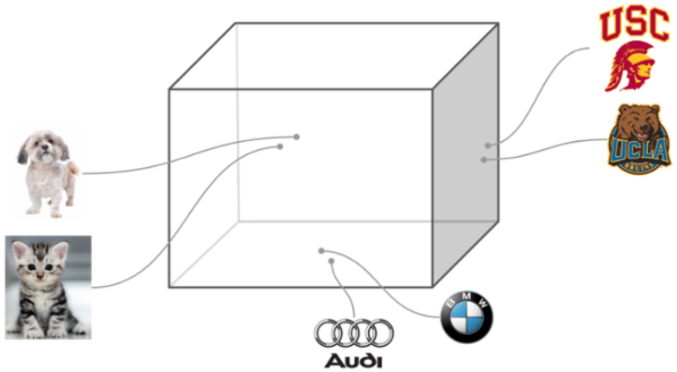
- Neural networks are easier for feature extraction.
- TextClassifier module can be modified for sentiment analysis.
- Neural networks generally have better performance, especially on QA tasks and when we lack data.
- Common parts can share for different AI modules.

Why Analytics Zoo for NLP?

- Analytics Zoo provides pipeline APIs, prebuilt models and use cases for NLP tasks.
- To provide practical experience for Azure big data users to build AI applications.
- Preinstalled image on Azure Marketplace for easy deployment.

Data Preprocessing

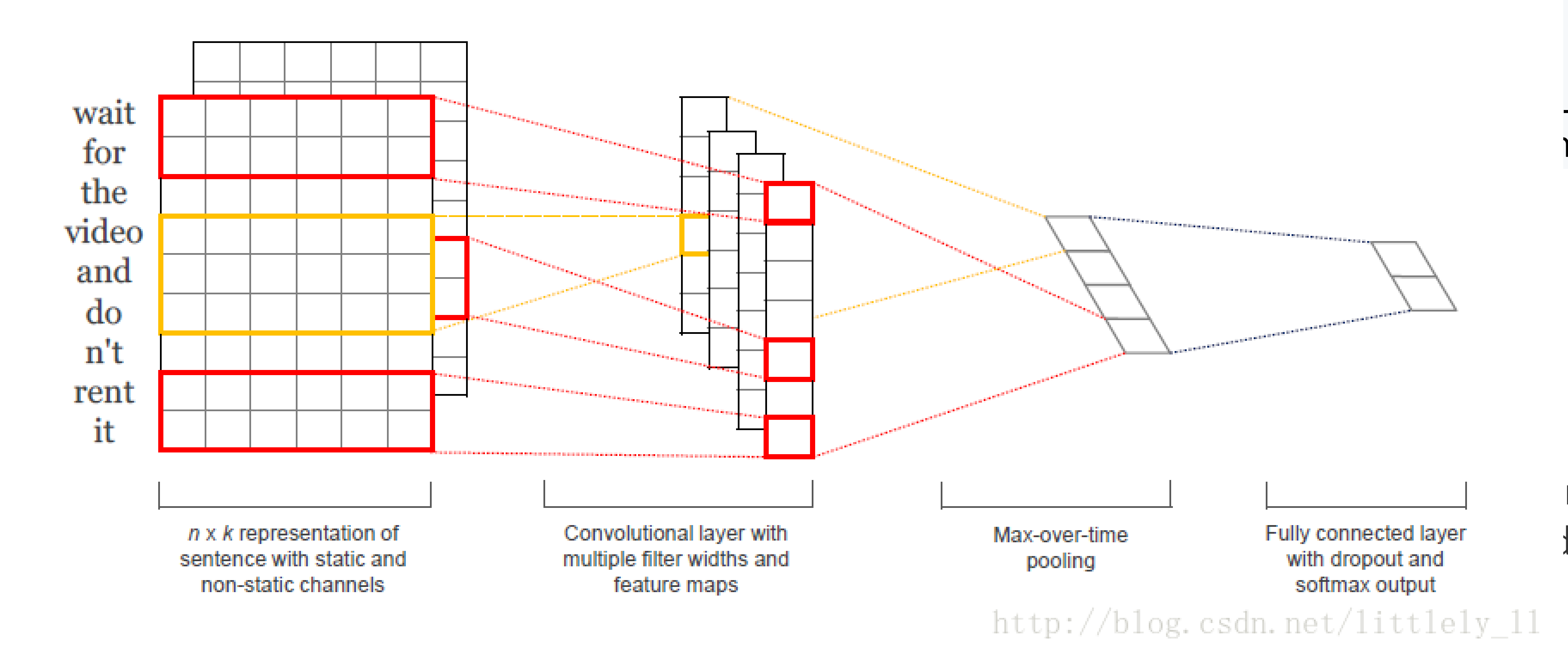
- Read cleaned text data as RDD where each record contains two columns (text, label).
- Common Steps
  - Tokenization: https://github.com/fxsjy/jieba
  - Stopwords removal
  - Sequence aligning
  - Word2Vec: https://github.com/facebookresearch/fastText
  - Conversion to BigDL Sample -> RDD[Sample]



Define TextCl
mode

from zoo.models.textclassification import TextClassifier

_length=500,
n=256)



r 'gru'.
ler.

*Photo from: https://blog.csdn.net/littlely_ll/article/details/79151403

Training, prediction and evaluation

Keras-Style API for distributed training:

```
text_classifier.compile(optimizer=Adagrad(learning_rate, decay),
                        loss="sparse_categorical_crossentropy",
                        metrics=["accuracy"])

text_classifier.set_checkpoint(path)
text_classifier.set_tensorboard(log_dir, app_name)

text_classifier.fit(train_rdd, batch_size=..., nb_epoch=..., validation_data=val_rdd)

text_classifier.save_model(model_path)

text_classifier.predict(test_rdd)
text_classifier.predict_classes(test_rdd)
```

Ways for improvement

- Check your data first (quality, quantity, etc.).
- Use custom dictionary for tokenization if necessary.
- Train word2vec for unknown words if necessary.
- Hyper parameters tuning (learning rate, etc.).
- Add character embedding, etc.

Service Integration

- Prediction service implemented in Java
- POJO-like API for low-latency local inference

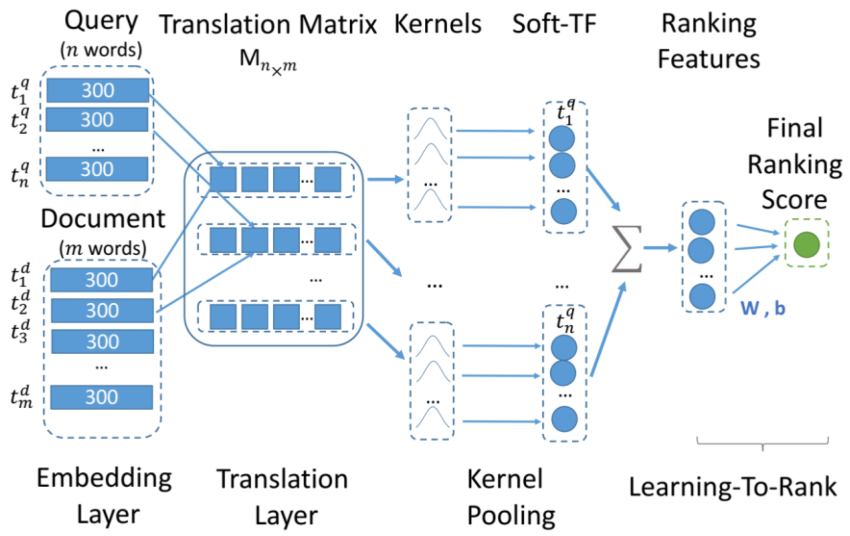
```
public class TextClassificationModel extends AbstractInferenceModel {
  public JTensor preProcess(String text) {
     //Preprocessing steps using Java API
  }
}

TextClassificationModel model = new TextClassificationModel();
model.load(path);
String sampleText = "text content";
JTensor input = model.preProcess(sampleText);
List<JTensor> inputList = new ArrayList<>();
inputList.add(input);
List<List<JTensor>> result = model.predict(inputList);
```

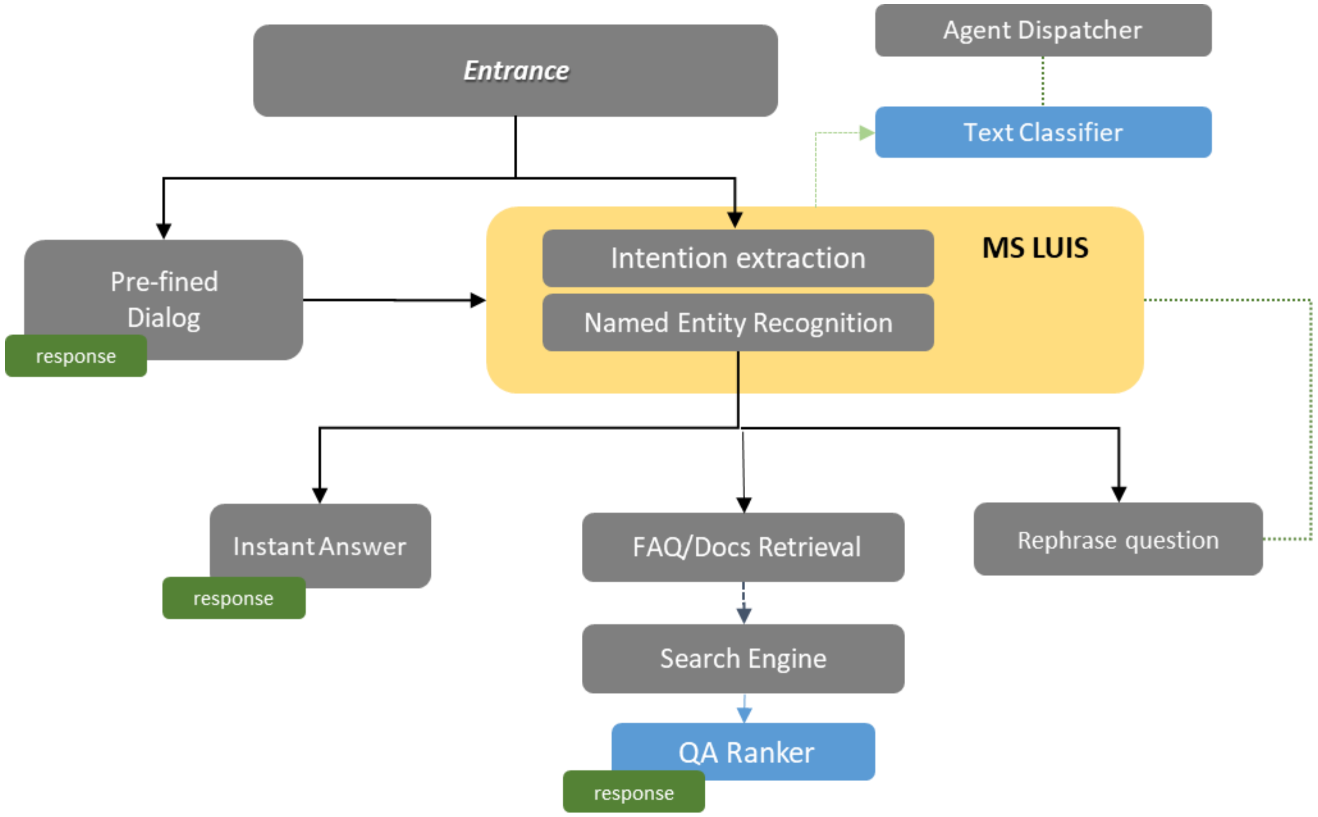
WebService example: https://github.com/intel-analytics/analytics-zoo/tree/master/apps/web-service-sample

QA Ranker

- Input: a query and a document pair.
- Similar preprocessing steps.
- Output: Relevance score or probability.
- Evaluation: Use normalized discounted cumulative gain (NDCG) or mean average precision (MAP).



Chat bot Flow



Analytics Zoo Implementation

- Easy to get: open source
- Easy to deploy compute env, just one click
- Easy to code: python / java
- Optimized for CPU and distribution
- Pre-defined neural networks

Model Management

- Online Feedback (automation)
  - Positive – useful answers; clicked links;
  - Negative – non-useful answers;
- Periodical Fine tune
  - Accumulated training material
  - Real user experience -> new benchmark
- Content Update
  - Re-index new Knowledge Base
  - Training set distribution
- Algorithm Update

Bot Demo

- WeChat: Microsoft云科技
- Webchat: https://support.azure.cn/zh-cn/support/support-azure/

Me:

虚拟机无法远程连接

Bot:

以下文章可能对您有帮助：

- Azure Windows虚拟机常见导致无法远程的操作
- Azure linux虚拟机常见导致无法远程的操作
- Linux 虚拟机虚拟网卡问题导致无法连接问题

Bot:

解决您的问题了么？

1. 解决了
2. 没有解决
3. 没有解决，转人工客服

Me: 输入您的问题 Send

virtual-machines/0.98882675, storage/0.003297496, virtual-network/0.003007766

Me:

如何进行ICP备案

Bot:

以下文章可能对您有帮助：

- ICP备案
- 首次备案
- 原备案在蓝云-备案变更

Bot:

解决您的问题了么?

1. 解决了
2. 没有解决
3. 没有解决，转人工客服

Me: 输入您的问题 Send

ICP/0.99798584, billing/3.540909E-4, automation/3.4101875E-4

Me:

Spark on HDInsight怎么用

Bot:

以下文章可能对您有帮助：

- DocumentDB API 性能级别
- Hadoop 组件和版本 - Azure HDInsight
- MySQL Database on Azure主从复制问题

Me:

Spark on HDInsight 怎么用

Bot:

以下文章可能对您有帮助：

- Hadoop 组件和版本 - Azure HDInsight
- Azure HDInsight 中的 Apache Spark 群集的已知问题
- Azure Toolkit for Eclipse：为 HDInsight Spark 创建 Scala 应用程序

Partner with us

- **GitHub: https://github.com/intel-analytics/analytics-zoo/**
- **Documents: https://analytics-zoo.github.io/**

- https://www.azure.cn/zh-cn/blog/2018/09/12/Using-Intel-Analytics-Zoo-to-inject-AI-into-customer-service-platform_PartI
- https://software.intel.com/en-us/articles/use-analytics-zoo-to-inject-ai-into-customer-service-platforms-on-microsoft-azure-part-1
- More blogs coming soon ☺