COTOBADESIGN

SageMakerによる
意図解釈サービス基盤の開発事例

2019-09-19

株式会社コトバデザイン
マネージャ/リサーチエンジニア
土田正明

COTOBA DESIGN

本日の内容

1. 会社紹介
2. 対話エージェント開発の概要
   - 「意図解釈サービス基盤」の必要性の導入として、対話エージェントの構成や作り方を簡単に説明します。
3. SageMakerを使った意図解釈サービス基盤の開発事例
   - 要件概要、アーキテクチャ、SageMakerの使い所や利用上の工夫の説明をします。

自己紹介

- 土田正明　博士（工学）
- 研究開発関連のキャリアが長く、主に自然言語処理を専門にしています。
- 最近の仕事内容
  - 自然言語処理関連の研究開発
  - [本日の話] 上記成果のためのサービス基盤の開発
  - 対話エージェントのオーサリングツール(Webアプリ)の開発

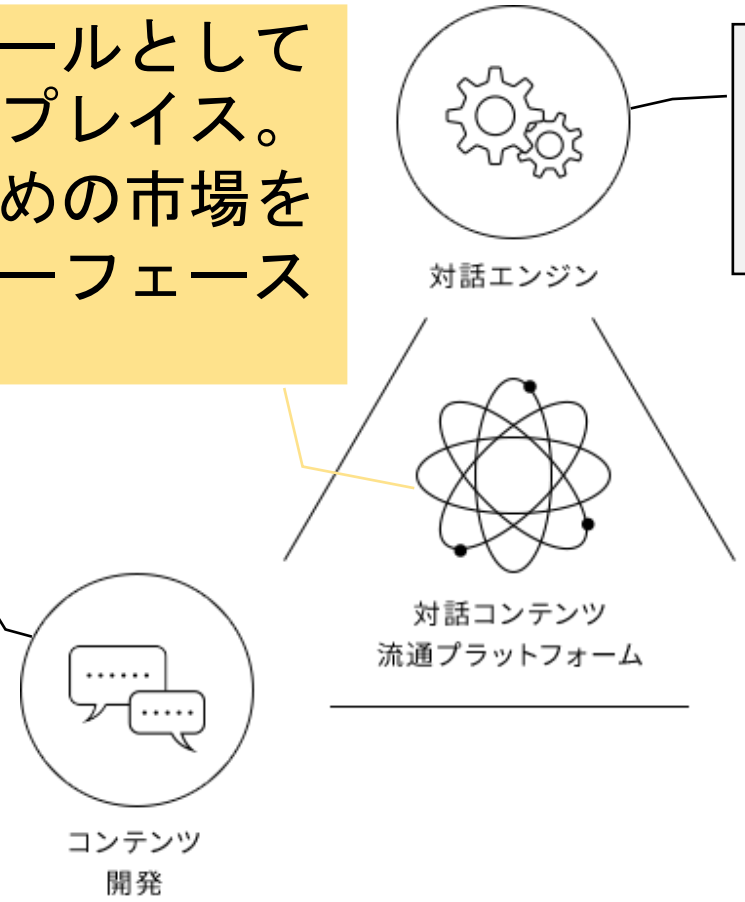
1. 会社紹介

コトバデザインはどんな会社？

Vision

自然言語インタフェースを変え，改良する個人の貢献によって，機械の知能がもたらす力を年齢，文化，性別，教育を超えて大衆に解放すること.

開発者成果物をモジュールとして流通させるマーケットプレイス。個人の貢献に報いるための市場を作り、自然言語インターフェースの社会実装を加速。



OSSにすることで、本エンジン用のコンテンツの実行環境構築が可能。実行環境をLOCKしない。

自分たち自身も開発し、対話エージェントのモジュールやベストプラクティスを提供。

誰もが対話エージェントのコンテンツを開発できるツールを提供。

最近の対外的アウトプット

2019年6月25日にCOTOBA Agentをクローズドβ公開

News Release
報道関係各位

COTOBADESIGN

2019年6月25日
株式会社コトバデザイン

対話エージェント実行環境 「COTOBA Agent」
クローズド・ベータ運用開始

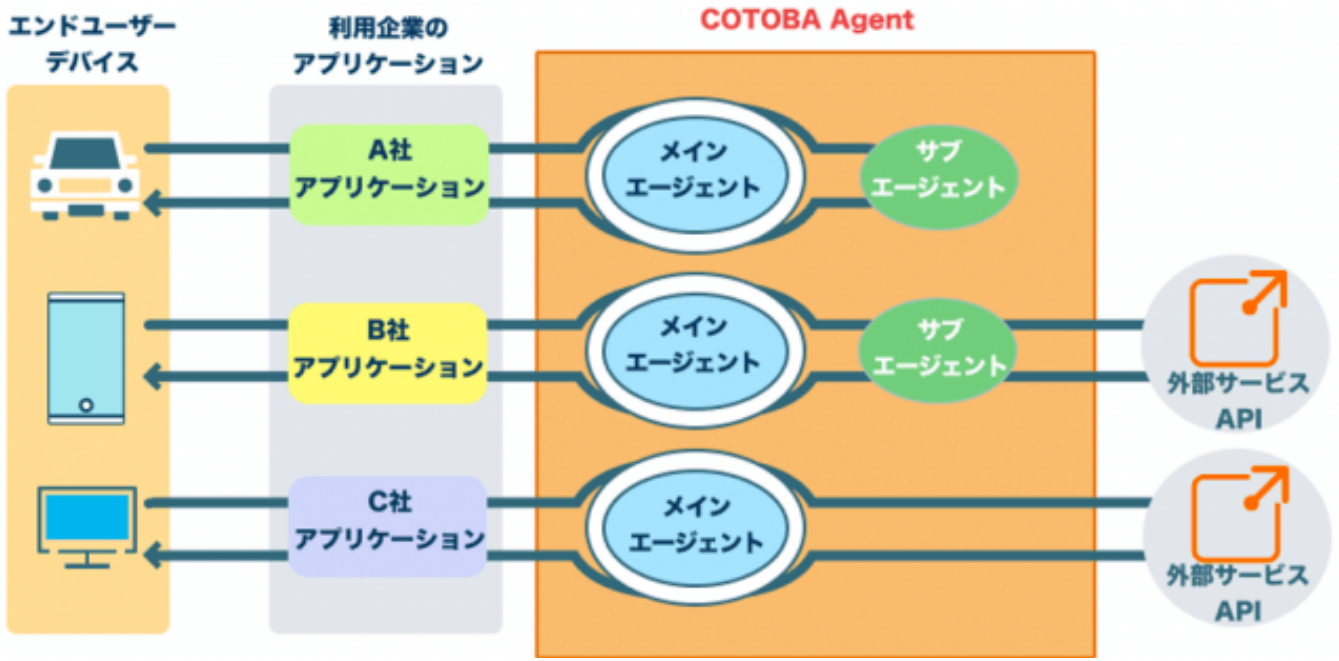
ポイント
- 対話エージェント実行環境 「COTOBA Agent」のクローズド・ベータ運用を開始。ユーザからのフィードバックを得て半年後に本格運用へ移行予定。
- 総務省からの委託研究成果である対話エンジンを核に、対話エージェントの開発・運用支援やセキュリティなどの実行環境機能を統合し、APIサービスとして提供。
- ハイブリッド構成の対話エンジンにより、音声アシスタント、企業・自治体の窓口業務の自動化、雑談対話などの多様な用途の対話エージェントが開発可能。
- ユーザからのフィードバックを受けて進化した対話エンジンは、オープンソース・ソフトウェアとして公開予定。クラウド、オンプレミスを問わずユーザの所望する計算機環境で、自由に対話エージェントの実装が可能。

株式会社コトバデザイン（本社：東京都渋谷区、執行役員社長CEO　栄藤稔）は、6月25日、対話エージェント実行環境 「COTOBA Agent」のクローズド・ベータ運用を開始しました。ユーザからのフィードバックを得て、半年後に本格運用へ移行する予定です。

COTOBA Agentは、総務省からの委託研究テーマ「高度対話エージェント技術の研究開発・実証」の成果である対話エンジンを核として、対話エージェントの開発・運用支援などを含む実行環境機能を統合し、API サービスとして提供されます。企業ユーザが実用的な対話エージェントを開発・運用する上で必須のセキュリティ、大規模複数エンドユーザの利用に耐えうるスケーラビリティも備えています。

https://cotobadesign.com/wpcontent/uploads/2019/06/COTOBA_Agent-20190625-3.pdf

対話的アプリケーションのためのバックエンド（エージェント）の開発と運用ができます。
本日はこの中の一部の開発事例のお話です。



We are hiring!!

- 以下に興味がある方、ご検討よろしくお願いします！
  - 対話エンジンの開発
  - 対話のための各種入力の認識系エンジンの開発
    - 自然言語、音声・音、画像、センサー情報、etc...
    - 認識すべき情報の企画や必要な訓練データの開発も含みます。
  - エンジンのウェブサービス化やそれらを柔軟にデプロイ・運用するためのサービス基盤の開発
  - 当社事業に興味のあるプロダクトマネージャ経験者
    - マーケティング、企画、事業開発を含みます。
- 関連するお問い合わせは下記からお願いししします！
  - https://cotobadesign.com/careers/
  - https://www.wantedly.com/companies/cotobadesign

2. 対話エージェント開発の概要

対話エージェントの大雑把な構成

パンケーキを食べられる店はあるかな？

意図解釈

インテント：レストラン検索
スロット
- 料理：パンケーキ
- エリア：？？

対話制御

レストラン検索

⋮

雑談応答検索

応答生成

エリアの希望はありますか？

質問:エリア

対話シナリオ（制御ルール）

・インテントがレストラン検索の場合で、料理とエリアが揃ったらレストラン検索を実行。不明なスロットがあれば質問。
・特定意図がなければ雑談応答検索を実行。

現在の状態と入力に基づき何をするかを決めるためのルール。

対話エージェントに合わせた意図解釈が必要

- 以下が密接に絡みあっているため、対話エージェントが違えば必要になる意図解釈機能が異なる。
  - 対話エージェントを通して提供する機能
  - 提供機能（例:レストラン検索）に必要なAPI
  - 要求に合わせて使う機能を決め、そのために必要な情報（例:料理, エリア）を認識するための意図解釈

- 特に、独自機能（API）を使う場合は既存の意図解釈機能の再利用は期待できないため、新規開発が必要。

ただし、意図解釈の開発も運用も簡単ではない

- 様々な発話を正しく認識できるようにする必要がある。
  - “パンケーキを食べたいんだけど、どこかないかな？”
  - “レストラン、パンケーキ、探して”
  - ”パンケーキを食べたいんだけど、どうやれば美味しくできるかな？“
  - “パンケーチを食べたいんだけど、我慢した方がいいのかな？”
- 認識ロジックをプログラムに落とすのは大変なことが多いので、近年では機械学習を使って作成する。
  - 「この入力をこう認識して欲しい」という例示から、そのように認識するモデルを機械に学習させる。
- 機械学習モジュールのサービス化やその安定運用は結構大変。
  - 各種機能のAPI化、モデル・データのバージョン管理、インフラ構築、デプロイ自動化、オートスケーリング、etc…

まとめると

対話エージェントの開発・運用には、意図解釈機能の開発・運用のサービス基盤も重要！！

3. SageMakerを使った意図解釈サービス基盤の開発事例

目的と設計方針

- 目的：個々の開発者に対し、機械学習で独自の意図解釈機能を作成し運用できるウェブサービスを提供する。
- 設計方針
  1. できるだけマネージドサービスを使う。
     - 運用工数に限りがあるのでサーバの保守をしたくない。
     - 可用性、耐障害性を意識せずに機能開発に集中したい。
  2. 機械学習の技術は日進月歩なので、特定の技術に依存しない作りにする。
     - 新技術が出た時の追加を容易にする

SageMakerはこの設計方針に合致 → 採用！

Amazon SageMaker

- 機械学習系マネージドサービス
  - 訓練: サーバ作成、リソース配置、訓練、訓練結果アップロード、サーバ削除を自動実行。
    - この仕組みを自分で作るのは結構面倒。
    - 訓練毎にサーバの作成と削除をするので無駄がない。
  - 推論 (運用) : エンドポイント作成、バックエンドの冗長構成、オートスケーリングをしてくれる。
- 機械学習の技術に非依存
  - SageMakerの取り決めに合わせてdocker imageを作れば訓練にも推論にも利用できる
    - 使う技術の追加＝その技術のdocker image化と登録

意図解釈モデルの訓練に使う２種類のデータ

訓練データ：このように認識して欲しいという例示

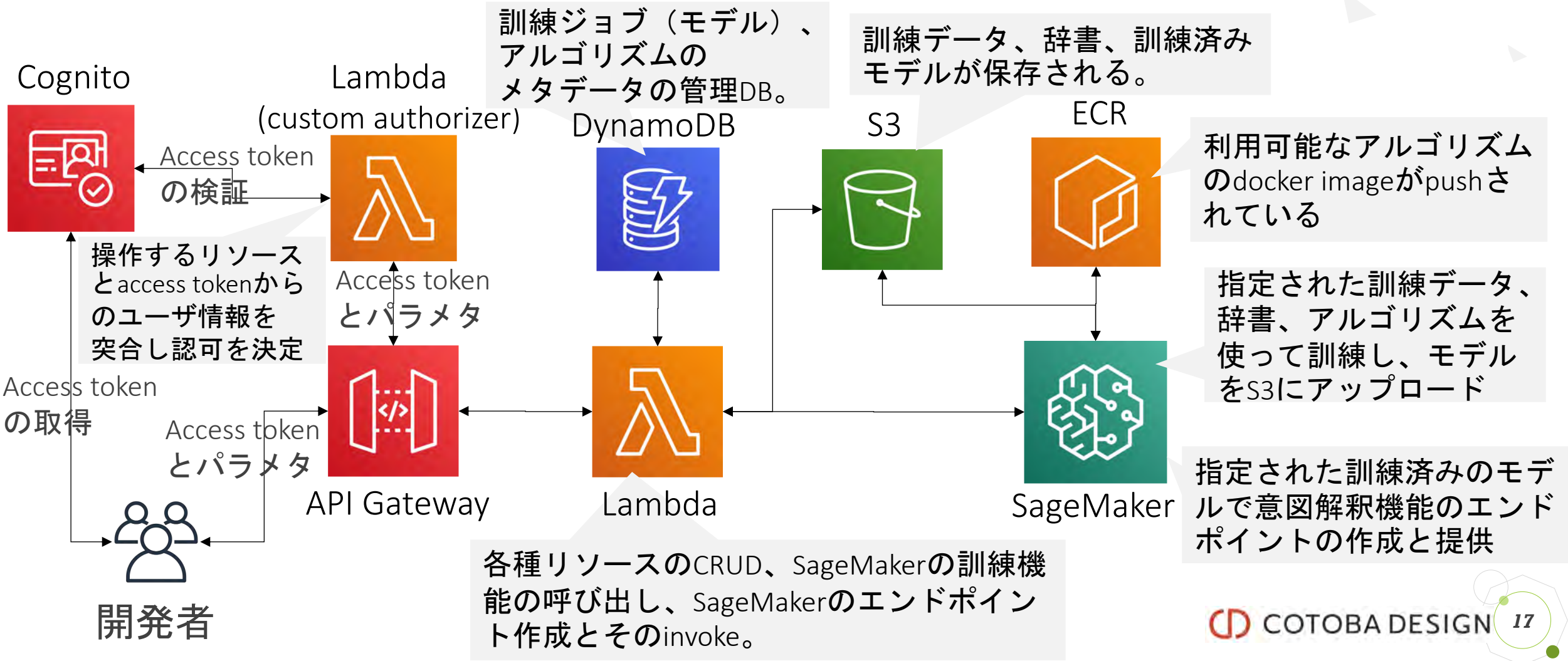
```
...
{
    “text”: “O型で1983年3月31日生まれの占いを教えて",
    "intent": ["占い"],
    "slot": [
     {
      "type": "血液型",
      "start": 0,
      "end": 2
     },
     {
      "type": "誕生日",
      "start": 3,
      "end": 13
     }
    ]
  },
...
```

辞書：訓練時に手がかり情報として利用。
例えば以下の辞書の場合、訓練データ中の血液型が”O”のみでも、”A”、”B”、”AB”も”O”と同種という手がかり情報を与えられるので、血液型として認識されやすくなる。

```
...
西舞鶴   電車新幹線駅
新浦安   電車新幹線駅
...
A        血液型
B        血液型
O        血液型
AB       血液型
...
```

意図解釈サービス基盤のアーキテクチャ（簡略化版）

SageMakerで利用するためのリソースのcreate, read, update, delete (CRUD)とSageMakerを呼ぶためのAPIを開発してサービス基盤化



基本的な意図解釈機能の作成と利用（CLIの利用例）

1. 訓練データリソースを作成（API経由でS3にアップロード）

```
$ cotoba nlu create-training-data --data-name train-data --file training_data.gz
```

2. （任意） 辞書リソースを作成（API経由でS3にアップロード）

```
$ cotoba nlu create-slot-dictionary --data-name slot-dic --file slot.dic.gz
```

3. アルゴリズム、訓練データ、辞書(任意)を指定してSageMakerの訓練ジョブを作成

この例ではアルゴリズムとそのハイパーパラメタはデフォルトとなる。

```
$ cotoba nlu create-model --training-data-id train-data --slot-dictionary-id slot-dic --job-name test-model
```

4. 訓練が正常終了したジョブでSageMakerのエンドポイントを作成

```
$ cotoba nlu create-endpoint --model-id test-model --slot-dictionary-id slot-dic
```

5. SageMakerのエンドポイントをinvoke （次スライド）

エンドポイント利用（curlでURLを叩く例）

URLマスキング

```
curl -s -X POST -H 'Content-Type:application/json' -H 'x-api-key:""' -d '{"utteranc
e": "渋谷から新大阪の乗り継ぎを知りたい"}' https:// | jq .
{
  "text": "渋谷から新大阪の乗り継ぎを知りたい",
  "intents": [
    {
      "intent": "乗り換え案内",
      "score": 0.9998147487640381
    }
  ],
  "slots": [
    {
      "slot": "電車新幹線駅(始点)",
      "startOffset": 0,
      "endOffset": 2,
      "entity": "渋谷",
      "score": 0.8669691093901105
    },
    {
      "slot": "電車新幹線駅(終点)",
      "startOffset": 4,
      "endOffset": 7,
      "entity": "新大阪",
      "score": 0.999360144158898
    }
  ]
}
```

URLマスキング

```
curl -s -X POST -H 'Content-Type:application/json' -H 'x-api-key:""' -d '{"utteranc
e": "渋谷から新大阪の乗り継ぎを知りたい人がいる"}' https:// | jq .
{
  "text": "渋谷から新大阪の乗り継ぎを知りたい人がいる",
  "intents": [
    {
      "intent": "others",
      "score": 0.9999759197235107
    }
  ],
  "slots": []
}
```

SageMaker利用で工夫したポイント(1/4)

- 訓練前のfail(‘Insufficient capacity error from EC2…’など）はイベントとしては検知できないので、訓練のFail系はDescribeTrainingJobの定期実行で監視
  - 管理しているメタ情報で「訓練中」となっているジョブのみが対象
  - 訓練の正常終了はS3へのmodel.tar.gzのcreateイベントでわかる

- 訓練のfailを検知できても単純にはリスタートできない（同じ訓練ジョブ名が使えずリスタートAPIもない）ので、枝番をつける形でリネームしてCreateTrainingJobを実行

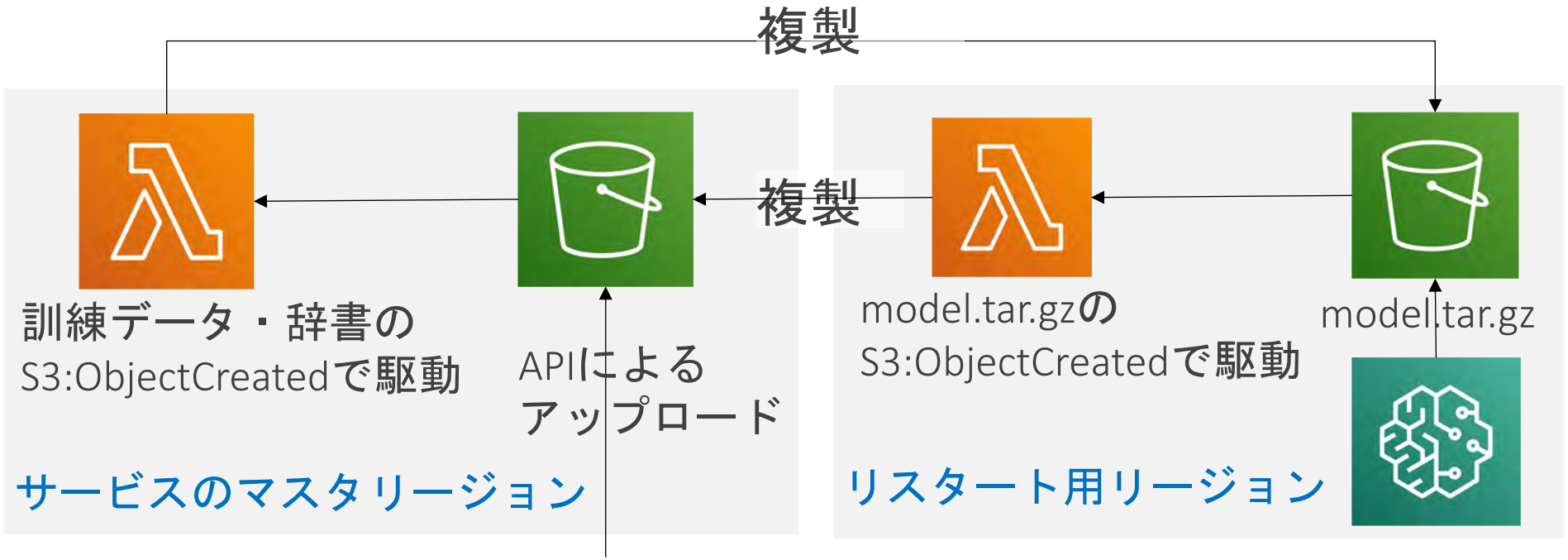
SageMaker利用で工夫したポイント(2/4)

- SageMakerの制限内でやりくりできるように、訓練ジョブをqueueにして同時実行ジョブ数をサービス全体で制御
  - 特定の開発者がたくさんジョブを登録できないように、同時にqueueに登録しておけるジョブ数は制限

- SageMakerのエンドポイント系の制限（同時に立ち上げられるエンドポイント数、エンドポイントに利用可能なインスタンス数）も全体で超えないように開発者毎に制限する仕組みを実装
  - リソース作成前に制限内であるかをチェックする

SageMaker利用で工夫したポイント(3/4)

- 希少なインスタンスタイプ(ml.p3.2xlarge)が枯渇('Insufficient capacity error from EC2...')することがあり、直ぐには空かないのでマルチリージョンでリスタートするための仕組みを実装。
  - S3、ECRはSageMakerと同リージョンでなければならないため、ECRの複製はデプロイ時に行い、S3のオブジェクトの複製はイベント駆動のLambdaで対応



SageMaker利用で工夫したポイント(4/4)

- エンドポイント作成で訓練時と異なる辞書を使うために、モデルアーティファクト（model.tar.gz）に辞書を入れ込む仕組みを実装。
  - モデルのパラメタはそのままで、入力となる特徴量（素性, feature）を調整する目的で辞書を変更するユースケースに対応。
  - CreateTrainingJobはInputDataConfigに指定して様々なファイルをdocker containerに渡せるが、CreateEndpointではできない。

訓練時

…
昭和　元号
平成　元号
…

エンドポイント作成時

…
昭和　元号
平成　元号
令和　元号
…

※「令和」が日付として認識されづらい問題が判明し、訓練データと辞書で元号の単語が日付になりやすい傾向が学習されている前提。

まとめ

- 対話エージェントが違えば必要な意図解釈機能も違う
- 意図解釈機能は機械学習で作成されることが多い
- そこで、開発者が独自の意図解釈機能を機械学習で作成・運用できるサービス基盤をSageMakerで開発
  - SageMakerをコアにラッパーを開発してサービス基盤化
  - 具体的には、SageMakerで利用するリソースのCRUDとSageMakerの機能を呼び出すためのAPI群を開発
- 不特定多数の開発者向けにSageMakerでサービス基盤を開発した事例として参考になれば幸いです

COTOBADESIGN