IWIbot Classifier

IWIbot Classifier ist ein Text Classification Module, das Neuronale Netzwerke benutzt. Es basiert auf den Algorithmen aus Text Classification using Neural Networks und ist Teil des IWIbot Projekts der Hochschule Karlsruhe. Das Classification Module wird als eine Python Flask App bereitgestellt und kann auf die IBM Cloud deployed werden.

Dieses Dokument soll dabei helfen die Entwicklungsumgebung für den IWIbot Classifier aufzusetzen und soll hilfreiche Tipps geben für die weitere Entwicklung.

Voraussetzungen

Folgende Komponenten werden benötigt: * IBM Cloud account * Cloud Foundry CLI * Git * Python * PyCharm (optional)

1. Clone die IWIbot App

Clone das IWIBot Repository und gehe in das Verzeichnis in dem sich der IWIBot Classifier befindet.

```
git clone https://github.com/HSKA-IWI-VSYS/IWIbot cd IWIbot/classification
```

2. Führe den Classifier lokal aus

Installiere die Abhängigkeiten die in der requirements.txt gelistet sind um den Classifier lokal ausführen zu können.

Es ist möglich optional eine virtuelle Umgebung zu erstellen um zu verhindern das die Abhängigkeiten mit anderen Python Projekten oder dem Betriebssystem kollidieren. `pip install -r requirements.txt`

Führe den Classifier aus. `python rest.py`

In PyCharm kann die **rest.py** auch einfach gestartet und debugged werden.

Eine Webtest-Oberfläche befindet sich unter: http://localhost:8000

3. Bereite den Classifier für das Deployment vor

Um in die IBM Cloud zu deployen, ist es wichtig das manifest.yml richtig zu konfigurieren. Der Classifier kommt mit einem bereits vorkonfigurierten Manifest.

Das manifest.yml beinhaltet Basis-Informationen über die zu den Classifier, wie den Namen in der Cloud, wie viel Speicher für pro Instanz alloziert werden soll

und die Route. Im aktuellen manifest.yml **random-route: false** generiert den Classifier immer unter der gleichen Route. Sollen mehrere Classifier Instanz deployen werden kann dies in **random-route: true** geändert werden und es wird eine zufällige Route erzeugt die einen Konflikt mit anderen verhindert. Es ist auch möglich **random-route: false** mit **host: IWIbotChosenHostName**, um einen Hostnamen selbst zu wählen. Weitere Informationen... `applications: - name: IWIBotClassifier random-route: false memory: 128M`

4. Deployment des Classifier

Es ist möglich den Classifier über die Cloud Foundry CLI zu deployen.

Wähle den passenden API Endpunkt `cf api <API-endpoint>`

Ersetze das *API-endpoint* im Befehl mit einem API endpoint aus der folgenden Liste.

URL	Region
https://api.ng.bluemix.net	US South
https://api.us-east.bluemix.net	US East
https://api.eu-de.bluemix.net	Germany
https://api.eu-gb.bluemix.net	United Kingdom
https://api.au-syd.bluemix.net	Sydney

Loge in deinen IBM Cloud Account

`cf login`

Aus dem *classification* Verzeichnis heraus pushe den Classifier in die IBM Cloud
`cf push IWIBotClassifier -b https://github.com/cloudfoundry/buildpack-python.git`
Der Classifier wird immer mit der aktuellen Python Version deployed in der **runtime.txt** kann die Python Version angepasst werden.

Das Deployen kann einige Minuten dauern. Kommt es zu Fehlern im Deployment Prozess kann mit dem Befehl `cf logs IWIBotClassifier --recent` der Fehler gesucht werden.

Nach abgeschlossenem Deployment sollte eine Meldung anzeigen das der Classifier läuft. Nun ist es möglich den Classifier unter der gelisteten URL in der Ausgabe des push Befehls anzusprechen. Es ist auch möglich den Befehl `cf apps` auszuführen um eine Übersicht über den Status des Classifiers einzusehen und um die URL zu sehen.

5. Eine Datenbank hinzufügen

Der Classifier benötigt eine NoSQL Datenbank zum persistieren seiner Daten. Aus diesem Grund wird der IBM Cloud ein Service hinzugefügt der diese bereitstellt und es findet die Konfiguration diese Datenbank in der IBM Cloud als auch lokal zu verwenden.

1. Öffne IBM Cloud in deinem Browser. Öffne das `Dashboard`. Wähle den Classifier aus durch klicken auf den Namen in der `Name` Spalte.
2. Wähle `Connections/Verbindungen` aus und dann `Connect new/Neue verbdinen`. (Optional kann auch eine bereits erstelle verbunden werden)
3. Im `Data & Analytics` Bereich, wähle `Cloudant NoSQL DB` und `Create` den Service.
4. Wähle `Restage` wenn gefragt wird. Die IBM Cloud wird den Classifier neu starten und die Datenbank Credentials für den Classifier bereitstellen mit der`VCAP_SERVICES` Umgebungsvariable. Diese Umgebungsvariable ist nur verfügbar wenn der Classifier auf der IBM Cloud ausgeführt wird.

Umgebungsvariablen ermöglichen es die Deployment Einstellungen vom Quellcode zu separieren, es ist also nicht nötig ein Datenbank Passwort im COde zu setzen, es ist auch mögliche diese Umgebungsvariable im Quellcode zu speichern.

6. Benutze die Datenbank lokal

Um den Classifier lokal auszuführen ist es nötig lokal eine Verbindung mit der Datenbank aufzubauen. Dafür wird eine JSON Datei die die Credentials für den zu benutzenden Datenbank Service speichert. Diese Datei wird NUR verwendet wenn der Classifier lokal ausgeführt wird. Falls es in der IBM Cloud ausgeführt wird, werden die Credentials aus der VCAP_SERVICES Umgebungsvariable gelesen.

1. Erstelle eine Datei mit Namen `vcap-local.json` im `IWIbotClassifier` Verzeichnis mit dem folgenden Inhalt: `{ "services": { "cloudantNoSQLDB": [ { "credentials": { "username":"CLOUDANT_DATABASE_USERNAME", "password":"CLOUDANT_DATABASE_PASSWORD", "host":"CLOUDANT_DATABASE_HOST" }, "label": "cloudantNoSQLDB" } ] } }`

2. In der IBM Cloud UI, wähle den Classifier -> Connections/Verbindungen -> Cloudant -> View Credentials

3. Kopiere und ersetze die `username`, `password`, und `host` aus den Credentials in das selbe Feld als in der `vcap-local.json` Datei und ersetze **CLOUDANT_DATABASE_USERNAME**, **CLOUDANT_DATABASE_PASSWORD**, und **CLOUDANT_DATABASE_URL**.

4. Führe den Classifier local aus. `python rest.py`

oder starte den Classifier in PyCharm.

Der Classifier kann unter: http://localhost:8000 getestet werden. Der Text der in das Feld eingeben wird, wird dann klassifiziert und das Resultat wird ausgegeben.

5. Mache die Änderungen die du willst und deploye erneut zur IBM Cloud! `cf push IWIBotClassifier -b https://github.com/cloudfoundry/buildpack-python.git`

Schaue die Instanz an unter der gelisteten URL in der Ausgabe des push Befehls, zum Beispiel, *iwibotclassifier.mybluemix.net*.

7. Datenbank füllen

Nach dem ersten Start des Classifiers ist die Datenbank noch leer zum Füllen der Datenbank nach dem ersten Start kann in der Web-Schnittstelle der Satz "populate" in das Textfeld eingegeben werden. Der Classifier initialisiert seine Trainer und Classifiers mit vorgegebenen Werten.

Warnung: abhängig von der gewählten Strategie der Datenbank ist es der Fall das es beim Füllen der Datenbank zu Fehlern kommt da nicht genügend Datenbank zugriffe pro Sekunde erlaubt sind, daher kann es notwendig sei mehrmals "populate" auszuführen. Die Demo-Seite quittiert nach erfolgreichem Initialisieren mit einem "POPULATED" Resultat.

8. Hilfreiche Links

- IBM-Cloud Get-Started-Python
- Cloudant Client Dokumentation
- Text Classification using Neural Networks

Aufbau

Der Classifier besteht aus drei Komponenten:, Trainer, Classifier und REST-Schnittstelle

- **Trainer:** Zuständig die verwaltung der Trainings-Daten für ein Neuronalen Netz (NN) und zum Training des NN basierend auf den gegebenen Trainings-Daten. Der Trainer speichert Trainings-Daten und NN in je einem Dokument in einer Datenbank. Das NN kann nach abschließenden Training und speichern vom Classifier aus der Datenbank geladen werden und darauf basierend eine Klassifizierung für einen Text geben.

- **Classifier:** Gibt eine Prognose über den Intend den der Satz beinhaltet basierend auf dem gegebenen NN des Trainers.
- **REST:** Ist die Schnittstelle des Classifiers nach außen Anwendungen können die Services die diese Schnittstelle bereitstellt aufrufen. Die Schnittstelle beinhaltet Endpunkte für das erhalten von Intends und Entities als auch Endpunkte zum modifizieren und trainieren von Neuronalen Netzen.

REST Schnittstelle

Endpunkte: * **/api/testIntent** Zum Testen des Classifiers durch die Web-Oberfläche.

- **/api/getIntent** Gibt den Intent eines Satzes zurück. Erwartet ein JSON im Request Body. Dieses JSON muss einen Satz beinhalten unter dem Schlüssel "sentence". Gibt als Response zurück den erhaltenen Request Body ohne den Satz und mit einer Classification, die einen Intent beinhaltet.
- **/api/getEntity** Gibt eine Entity eines Satzes zurück basierend auf den mitgegebenen vorherigen Intent. Erwartet ein JSON im Request Body. Dieses JSON muss einen Satz beinhalten unter dem Schlüssel "sentence" und einen vorherigen Intent unter dem Pfad "/context/priorIntent/intent". Gibt als Response zurück den erhaltenen Request Body ohne den Satz und mit einer Classification, die eine Entity beinhaltet.
- **/api/addIntent** Fügt einen Satz mit dem entsprechenden Intent dem Classifier für die Intents hinzu. Erwartet ein JSOn mit dem Schlüssel "sentence" für den hinzuzufügenden Satz und "intent" für den dazu gewollten Intent
- **/api/trainIntents** Trainiert den Classifier für die Intents neu.
- **/api/addIntent** Fügt einen Satz mit der entsprechenden Entity zum entsprechenden Classifier für die Entities des übergebenen Intents hinzu. Erwartet ein JSOn mit dem Schlüssel "sentence" für den hinzuzufügenden Satz, "entity" für die gewollte Entity und "intent" für den entsprechenden Intent, zu dem diese Entity gehört.
- **/api/trainEntity** Trainiert den Classifier für die Entities zu einem Intents neu. Erwartet ein JSON mit dem "intent" für den die Entities neu trainiert werden sollen.

Ausblick

Im Rahmen des Projekts wurden auch schon der Grundstein gelegt den Classifier zu erweitern. Folgende Ideen sind Teilweise schon umgesetzt:

Der Classifier kann schon erweitert werden, der nächste logische Schritt wäre eine Strategie zu entwickeln wie Feedback von Nutzern in die Trainings-Daten eingeführt werden kann, dabei kommen 2 möglichkeiten in Frage:

- **Threshold:** Wen ein Nutzer eine Anfrage stellt und das Resultat negativ ist kann der Nutzer eine Antwort geben welches Resultat den Richtig wäre, geben viele Nutzer zu einem Satz das gleiche Endresultat als Feedback kann dieses den Trainings-Daten hinzugefügt werden und der Trainer neu trainiert werden.
- **Trusted Sources:** Bei dieser Variante gibt der Nutzer Nutzer zwar auch Feedback dieses wird aber erst übernommen wenn eine vertraute Person dieses auch verifiziert. Es ist auch möglich einige Nutzer aus vertraut einzustufen und ihr Feedback sofort zu übernehmen. Der Trainer besitzt bereits ein trusted flag beim Hinzufügen von Trainings-Daten, die Logik dafür ist aber noch nicht Implementiert.