Up-to-Speed

Data Science Task

This is a suggested first task for all new team members joining the team in a Data Scientist position. After the successful implementation of all the requirements the new team member will have basic knowledge on some of the tools used by the NLP team at TDX.

The goal is to create a REST service in Python that performs some basic NLP tasks for a given input using 3rd-party libraries and a Python library that allows transparent communication with the server.

Component #1

The server should perform the following operations:

- Tokenization: split a phrase into tokens.
- Stop Word Identification: identify stop words tokens in a phrase.
- Part-of-Speech Tagging: identify PoS Tags for all tokens in a phrase.
- Dependency Parsing: return a tree with dependency relations between each token.
- Name Entity Recognition (NER): identify named entity tokens (e.g. Persons, Countries, etc).
- Get closest words using word embeddings: for a given token or phrase return the closest tokens using word embeddings techniques (e.g. fastText).

It is up to you to choose the following:

- Which libraries to use (be careful and always read the LICENSE of each 3rd-party library - some doesn't allow commercial usage).
  - Some examples of libraries that we use are: NLTK, spaCy, gensim, TextBlob, Duckling, ...
- Which data structures will be used for input and output.

Requirements:

- Implement the operations above.
- Expose a REST API (using JSON) and implemented in Flask.
- Implement application logging.
- Implement basic unit tests.
- Use Python 3.
- Optional: run using a Docker container.

Component #2

A python library that will communicate with the server (Component #1). This will allow anyone to use the server without having to handle JSON marshalling, unmarshalling and HTTP connection details.

Requirements:

- Should give access to all the operations that the server supports.
- Use an existent REST client (no need to implement one).
- Should have a clean API and hide the details bellow (e.g.: the user should be aware that this library is a HTTP client).
- Implement application logging.
- Implement basic unit tests.
- Use Python 3.

Backend Task

This is a suggested first task for all new team members joining the team in a Backend Engineer position. After the successful implementation of all the requirements the new team member will have basic knowledge on some of the tools used by the NLP team at TDX.

The goal is to create a REST service that returns Part-of-Speech Tags (PoS) for a given text input using a 3rd-party library (spaCy), a CLI to consume the API endpoints, and logging all requests to a database (e.g. for the phrase "How are you today?", the system should return a JSON object with the following information: How/ADV, are/VERB, you/PRON, today?/NOUN).

Component #1

Java micro-service using Quarkus and JAX-RS. You can follow the Quarkus Getting Started Guide, step 4, to create the skeleton app.

Requirements

- Expose a REST API (using JSON) that will receive the text to analyze and return the PoS Tags.
- Process the PoS Tagging for the request, provided by Component #2. [TIP: https://quarkus.io/guides/rest-client-guide]
- Register all requests in the Database, provided by Component #3, using JPA and in an asynchronous way. [TIP: https://quarkus.io/guides/hibernate-orm-guide]
- Format for each JSON request and response are up to you but in the future it is important to coordinate with the frontend team.
- Implement application logging using SLF4J.
- Implement basic JUnit tests and run them using the IDE and also Maven.
- Make use of Maven.
- Run using a Docker container.

Component #2

Python micro-service that will perform the PoS Tagging using spaCy.

Requirements

- Run using a Docker container.
- You can search for a ready to used image at hub.docker.com (be careful and always read the LICENSE of each 3rd-party library - some doesn't allow commercial usage).

Component #3

Database using PostgreSQL.

Requirements

- Run using a Docker container.
- You can search for a ready to used image at hub.docker.com.
- A SQL file should run when the container starts to create the logging table.

Component #4

Java CLI tool that will connect to Component #1 using HTTP.

Requirements

- Allow the user to pass a sentence as argument (e.g. java -jar my-cli.jar -s "How are you today?").
- Allow the user to set the Host and Port using arguments (e.g.: -h xpto.com -p 8080)
- Output to the stdout the PoS tags for the given text.
- Make use of Maven.
- Run using a Docker container.

General Requirements

- Create a local Git repository for this exercise.
- Each component should have his own directory.
- Component #1, #2 and #3 should be used in a docker-compose setup.