Natural Language Processing with TextSpace

Note

This is the documentation for version 1.2.0 of TextSpace.

TextSpace is an Natural Language Processing (NLP) service for intent classification and entity extraction. For example, TextSpace takes a sentence like

```
"I wish to listen to a new song by Green Day on Spotify"
```

and returns structured data like

```
{
  "intent": "play_media",
  "entities": {
    "app" : "spotify",
    "artist" : "green day"
  }
}
```

It has been developed for multiple usecases like, Chatbots, Recommender Systems, Information Indexing and Retrieval, Research, etc. The target audience is

- **People who have less/no knowledge about NLP and Machine Learning**, but they know their usecase well and know how NLP can help them in solving their problems. TextSpace can act as the perfect black-box for this group.
- **People who have minimal/good knowledge about NLP and Machine Learning (ML)**, and they want to configure existing models, or add their own for their specific usecase.

You can use TextSpace as a replacement for wit , LUIS , or Dialogflow, the only change in your code is to send requests to `localhost/your_ip` instead.

Why might you use TextSpace instead of one of those services?

- you don't have to hand over your data to FB/MSFT/GOOG
- you don't have to make an `https` call every time.
- you can tune models to work well on your particular usecase.

TextSpace has a concept called **Pipelines**, which are NLP pipelines composed of multiple components. TextSpace comes with pre-built pipelines for ***Intent Classification***, ***Entity**

Extraction*, and both together. More details can be found in Processing Pipeline.

Supported Languages

TextSpace speakes **36 languages**, which are

- Arabic (ar)
- Danish (da)
- German (de)
- Greek (el)
- English (en)
- Spanish (es)
- Finnish (fi)
- French (fr)
- Irish (ga)
- Hebrew (he)
- Croatian (hr)
- Hungarian (hu)
- Icelandic (is)
- Italian (it)
- Georgian (ka)
- Korean (ko)
- Norwegian (nb)
- Dutch (nl)
- Polish (pl)
- Portuguese (pt)
- Romanian (ro)
- Swedish (sv)
- Ukrainian (uk)
- Vietnamese (vi)
- Chinese (zh)
- Hindi (hi)
- Bengali (bn)
- Assamese (as)
- Tamil (ta)
- Telugu (te)
- Kannada (kn)
- Malayalam (ml)
- Gujarati (gu)
- Marathi (mr)
- Punjabi (pu)
- Odia (or)

The code inside the brackets are the ISO codes for the respective languages. We will refer to these codes throughout the documentation. For more details about the languages supported refer to Language Support.

Quickstart

Prerequisites

- Python 3.5/3.6
- Java8+
- (Tested on) Ubuntu 16.04 or higher
- docker
- docker-compose

Docker

Follow the instructions below to get TextSpace up and running

```
$ cd <project_root_dir>
$ docker-compose -f docker/compose_files/docker-compose.yml up
$ curl -X POST   'http://localhost:5000/train?token=0123456789&language=en' --header
"Content-Type: application/json" -d @data/datasets/demo/training_data.json
{
  "info": "new model trained: model_20190114-000545"
}
$ curl -X POST localhost:5000/parse?token=0123456789 -d '{"q":"hello"}' | python -m
json.tool
{
    "intent": {
        "name": "greet",
        "confidence": 0.992779961448352
    },
    "entities": [],
    "fsd": 0.9885401414535943,
    "intent_ranking": [
        {
            "name": "greet",
            "confidence": 0.992779961448352
        },
        {
            "name": "goodbye",
            "confidence": 0.004239819994757724
        },
        {
            "name": "set_alarm",
            "confidence": 0.002111400060032927
        },
        {
            "name": "play_media",
            "confidence": 0.0008688184968572928
        }
    ],
    "text": "hello"
}
```

There you go! you just parsed some text. Next step, do the Tutorial: A Simple Smart Home Assistant.

Here you trained a TextSpace model in English. You can do the same for all the other supported language just by changing `en` to the respective language code.

> **Note**
>
> This demo uses a very limited ML model. To apply TextSpace to your use case, you need to train your own model! Follow the tutorial to get to know how to apply TextSpace to your data.

About

TextSpace is a Natural Language Processing service, which has APIs for building your own language parser using state of the art NLP and ML models.

TextSpace is written in Python, but it you can use it from any language by Using TextSpace as an HTTP server. If your project *is* written in Python you can simply import the relevant classes.

TextSpace has been developed by NeuralSpace, and is built upon Rasa NLU. It uses multiple open-source libraries/packages. But most importantly, it uses Scikit-learn, Polyglot and Duckling.

Memory Footprint & Training Time

Despite supporting 36 Languages, TextSpace can be used in all of them in real-time. TextSpace has a memory foot-print of **50-150MB** per model. The amount of memory TextSpace takes depends on how big the Polyglot model is for that language. Not all languages use polyglot, and not all polyglot models are huge.

- On **small datasets** (4 intents, 10-12 samples per intent), TextSpace takes **1-2 minutes to train** and a **few seconds to serve** a trained model.
- On **large datasets** (130 intents, 30 samples per intent), TextSpace takes **25 minutes (small gridsearch) to train**, and **a few seconds to serve** the trained model.