Named Entity Recognition, Linking and Generation for Greek Legislation

Introduction

This project contains the evaluation, prediction and training of the neural networks proposed in the paper "Named Entity Recognition, Linking and Generation for Greek Legislation" as well as the models themselves.

Files

The dataset contains 3 main folders:

- The **VECTORS** folder contains the word embeddings we trained for W2V based on a big corpus of greek legislation documents.
- The **PYTHON** folder contains sample code for our approach in two .py files (we also include the code of our proposed models). Once you have implemented your own Dataset Loader and load your data into the neural network and depending on your parameters' calibration, the network of your choice will be trained, fitted and then evaluated, providing a thorough classification report in a txt file.
- The **DATASETS** folder contains 2 subfolders:
  - The **ENTITY RECOGNITION** folder contains the annotated dataset we used to evaluate our **NER** component. It is already split into **TEST**, **TRAIN** and **VALIDATION**. Each of these folders contains the .txt (original document) and .ann (files with the annotations and offsets they are found in the text) files .
  - The **ENTITY LINKING** folder contains csv files with the entities we extracted for the classes **PERSON**, **GPE** and **LEGISLATION REFERENCE** as well as the target links found by **Silk** for them on the third-datasets we focused on (Kallikratis-GAG and DBpedia Greek Politicians).
- The **RDF** folder contains the RDF datasets we produced with our workflow. DBpedia Greek Politicians, GAG (Kallikratis), structured legislation documents in RDF format and our own entities found and recognized throughout the text. *Please note that we include both greek and english labels for all rdfs:labels to make queries easier.*

Usage

- **VECTORS**
  - **W2V_10_100.bin** contains the actual embeddings we used for our experiments.
- **PYTHON**
  - **keras_models.py** contains the full code of our proposed networks so that one can experiment and/or validate their performance.
  - **experiment_ner.py** contains sample code that pipelines the loading of **Word2Vec** word embeddings and the actual input for tests, conducting grid search for a set of configuration the user needs, providing an evaluation report for all experiments.
- **DATASETS**
  - **ENTITY RECOGNITION** In order to feed labelled entities for supervised learning into our neural nets, we require that data in a convenient format. As explained in the paper, we used **brat** to annotate a set of 254 greek legislation documents (the .txt files), obtaining an annotation file (.ann) for each document. An annotation file contains lines with the format **ID LABEL START END STRING**. **ID** is the **id** of the annotation of the text, **LABEL** is the annotation class (the class that entity was classified as), **START** and **END** are the offsets the entity exists in the text and **STRING** is the actualy entity as a string. Please keep in mind that some potential irregularities in the .txt files (such as greek typos) are actually due to the documents of greek legislation existing only in PDF format. Therefore, the .txt files provided are originated from converting PDF files.

    This folder includes the following:

    i) **TEST** contains the test part of our dataset. It has two main folders containing the .txt and corresponding .ann files for each legal document. The documents are also in folders that indicate their issue/class and year of publication.

    ii) **TRAIN** contains the train part of our dataset. It has two main folders containing the .txt and corresponding .ann files for each legal document. The documents are also in folders that indicate their issue/class and year of publication.

iii) **VALIDATION** contains the validation part of our dataset. It has two main folders containing the .txt and corresponding .ann files for each legal document. The documents are also in folders that indicate their issue/class and year of publication.

– **ENTITY LINKING** We provide the csv files with the links we discovered with **Silk** as a benchmark for any future evaluation/comparison with our approach. That is particularly useful because a comparing experiment can validate which links it found correctly, which it did not and which were "close" enough.

This folder includes the following:

i) **gpe_linking_test.csv** contains the **GPE** entities we found inside the test part of our dataset. It contains the columns "Original Label" (the string of the entity exactly as it was found inside the text), "Processed Label" (the resulting label after processing with original label with heuristic rules), "Target Label" (the label of the matching entity **Silk** produced from the **GAG/kallikratis dataset**) and "Target URI" which is the URI of the target entity. Obviously entities that could not meet **Silk**'s matching criteria do not have a "Target Label" or "Target URI".

ii) **person_linking_test.csv** contains the **PERSON** entities we found inside the test part of our dataset. It contains the columns "Original Label" (the string of the entity exactly as it was found inside the text), "Processed Label" (the resulting label after processing with original label with heuristic rules), "Target Label" (the label of the matching entity **Silk** produced from the **DBpedia Greek Politicians dataset**) and "Target URI" which is the URI of the target entity. Obviously entities that could not meet **Silk**'s matching criteria do not have a "Target Label" or "Target URI". Inspecting the "Original Label" column shows the problem with having good matches for persons, as most full names have the first name with an initial, producing wrong links for politicians with the same surname etc.

iii) **leg_ref_test.csv** contains the **LEGISLATION REFERENCE** entities we found inside the test part of our dataset. It contains the columns "Original Label" (the string of the entity exactly as it was found inside the text), "Processed Label" (the resulting label after processing with original label with heuristic rules), and "Target URI" which is the URI of the entity produced utilizing heuristic rules. There is no need to conduct a **Silk** interlinking task for legislation references, because the links we are generat-

ing are following the **ELI** specification and therefore the resulting link is either valid or invalid straight from its generation.

- **RDF**
  - **DBpedia** contains the Greek Persons we extracted from DBpedia as well as some basic information about them (depiction, summary, birth date, birth place, political party, political party affiliation).
  - **Entities** contains all the RDF information involving entities extracted from the legislation text by our state-of-the-art BILSTM-BILSTM-LR and their interlinking with the other datasets via **Silk**.
  - **GAG** contains the GAG (Kallikratis dataset).
  - **Legislation** contains 150,000 issues of the Greek Government Gazette in the period of 1990-2017 encoded in RDF format and also structured to their fundamental components (up to linea).

Ideally, we would wish to interlink the other extracted entity types (organizations, geographical landmarks, public documents references) with third-party datasets as well. However, there are not any sufficient greek datasets with relevant information of use and therefore we could not interlink these entities.

Sample Code

The entities extracted by our models belong to one of the following categories:

- **Person**. Any formal name of a person mentioned in the text. There are most probably Greek government members.
- **Geopolitical Entity**. Any reference to a geopolitical entity (e.g., country, city, Greek administrative unit, etc.).
- **Organization**. Any reference to a public or private organization, such as: international organizations (e.g., European Union, United Nations, etc.), Greek public organizations (e.g., Social Insurance Institution) or private organizations (e.g., companies, NGOs, etc.).
- **Geographical Landmark**. References to geographical entities such as local districts, roads, farms, beaches, which are mainly included in legislation related to topographical procedures and urban planning.

- **Legislation Reference**. Any reference to Greek or European legislation (e.g., Presidential Decrees, Laws, Decisions, EU Regulations and Directives etc.).
- **Public Document Reference**. Any reference to documents or decisions that have been published by a public institution (organization) that are not considered a primary source of legislation (e.g., local decisions, announcements, memorandums, directives).

In order to successfully run the evaluation, the following dependencies are required by Python:

- **Python 3**, strongly recommending Anaconda 3 since it comes with many libraries pre-installed.
- **tensorflow**, a Python library for neural networks, ideally tensorflow-gpu if you have an NVIDIA(tm) graphics card to exploit speedups from CUDA.
- **keras**, a Python library that uses tensorflow as a backend and augments the library's usability.
- **sklearn**, for the functions that evaluate a neural network.
- **gensim**, in order to load the word embeddings.

As explained before, however, the user needs to provide a Dataset Loader for the code to function as expected. When that is provided, the ENTITY RECOGNITION data (both .txt and corresponding .ann file for each document) should be provided alongside the word embeddings from the VECTORS folder. Feeding all that and choosing a set of configurations will run these configurations, providing an evaluation report as a last step.

Authors

- Ilias Chalkidis - ilias.chalkidis
- Iosif Angelidis - metimdjai

License