PPI challenge

Readme file

TEAM FORMATION

Name: Luca Becchetti
Email: becchetti@diag.uniroma1.it
Affiliation: Department of Computer, Control, and Management Engineering “Antonio Ruberti”, Sapienza University of Rome.
Contribution: Team leader.

Name: Adriano Fazzone
Email: fazzone@diag.uniroma1.it
Affiliation: Department of Computer, Control, and Management Engineering “Antonio Ruberti”, Sapienza University of Rome.
Contribution: Software Engineer and R&D of topological/biological features and Analysis of Results

Name: Leonardo Martini
Email: martini@diag.uniroma1.it
Affiliation: Department of Computer, Control, and Management Engineering “Antonio Ruberti”, Sapienza University of Rome.
Contribution: R&D of topological/biological features and Analysis of Results

CODE DESCRIPTION/USAGE

The root Folder `PPI_Challenge_submission_Becchetti_Fazzone_Martini` contains 3 directories that are related:

```
code             : source code written in Python3 and Java.
datasets         : external dataset used by the method.
submitted_results: results from internal and external validation.
```

Notice that dataset paths are hard coded in the python files that you can find in the Source Folder. Please, don’t move these folders otherwise the program will stop working!

For performing the **Topological-only-Internal evaluation**:

..) Run the Topological-Feature extractor tool from `PPI_Challenge_submission_Becchetti_Fazzone_Martini/code/topological_feature_extractor/bin` in the following way:

```
    ..) $ java -Xms12g -Xmx12g algos.TopologicalFeaturesExtractor <i|e> <directory_containing_the_input_file> <input_PPI_file_name> <output_EXISTING_directory_that_will_contain_all_the_10_TRAINING_and_VALIDATION_sets>
    ..) Example:
        $ java -Xms12g -Xmx12g algos.TopologicalFeaturesExtractor i /Users/ikki/Dropbox/PPI/PPIChallenge/datasets/PPI_challenge_official_datasets HuRI.csv
```

/Users/ikki/Dropbox/PPI/PPIChallenge/datasets/PPI_challenge_official_datasets/HuRI__TRAINIG_and_VALIDATION

..) Run the Internal-Validator tool from \/code/ranker_and_evaluator in the following way:

    ..) $ python3 topological_internal_validation.py
<directory_that_contains_all_the_**10_TRAINING_and_VALIDATION_sets**>
<input_PPI_file_name>
<output_EXISTING_directory_that_will_contain_the_**CROSS_VALIDATION_RESULTS_FILE**>

For performing the **Topological-and-Biological-<u>Internal</u> evaluation**:

..) Run the Topological-Feature extractor tool from
PPI_Challenge_submission_Becchetti_Fazzone_Martini/code/topological_feature_extractor/bin in the following way:

    ..) $ java -Xms12g -Xmx12g algos.TopologicalFeaturesExtractor <i|e>
<directory_containing_the_input_file> <input_PPI_file_name>
<output_EXISTING_directory_that_will_contain_all_the_**10_TRAINING_and_VALIDATION_sets**>

    ..) Example:

        $ java -Xms12g -Xmx12g algos.TopologicalFeaturesExtractor **i**
/Users/ikki/Dropbox/PPI/PPIChallenge/datasets/PPI_challenge_official_datasets
HuRI.csv
/Users/ikki/Dropbox/PPI/PPIChallenge/datasets/PPI_challenge_official_datasets/HuRI__TRAINIG_and_VALIDATION

..) Run the Biological-Feature extractor tool from
PPI_Challenge_submission_Becchetti_Fazzone_Martini/code/biological_feature_extractor in the following way:

    ..) $ python3 biological_feature_extractor.py
<directory_that_contains_all_the_**10_TRAINING_and_VALIDATION_sets**>
<output_EXISTING_directory_that_will_contain_all_the_**10_TRAINING_and_VALIDATION_sets_INTEGRATED_WITH_BIOLOGICAL_FEATURES**>

..) Run the Internal-Validator tool from
PPI_Challenge_submission_Becchetti_Fazzone_Martini/code/ranker_and_evaluator in the following way:

    ..) $ python3 biological_and_topological_internal_validation.py
<directory_that_contains_all_the_**10_TRAINING_and_VALIDATION_sets_INTEGRATED_WITH_BIOLOGICAL_FEATURES**> <input_PPI_file_name>
<output_EXISTING_directory_that_will_contain_the_**CROSS_VALIDATION_RESULTS_FILE**>

For performing the **Topological-only-<u>External</u> evaluation**:

..) Run the Topological-Feature extractor tool from
PPI_Challenge_submission_Becchetti_Fazzone_Martini/code/topological_feature_extractor/bin in the following way:

    ..) $ java -Xms12g -Xmx12g algos.TopologicalFeaturesExtractor <i|e>
<directory_containing_the_input_file> <input_PPI_file_name>
<output_EXISTING_directory_that_will_contain_the_**ENTIRE_TEST_set**>

    ..) Example:

```
        $ java -Xms12g -Xmx12g algos.TopologicalFeaturesExtractor e
/Users/ikki/Dropbox/PPI/PPIChallenge/datasets/PPI_challenge_official_datasets
HuRI.csv
/Users/ikki/Dropbox/PPI/PPIChallenge/datasets/PPI_challenge_official_datasets/HuRI_
_TEST
```

..) Run the External-Validator tool from `PPI_Challenge_submission_Becchetti_Fazzone_Martini/code/ranker_and_evaluator` in the following way:

```
    ..) $ python3 topological_external_validation.py
<directory_that_contains_the_ENTIRE_TEST_set>
<output_EXISTING_directory_that_will_contain_the_FIRST_500_PREDICTED_NON_INTERACTING
_PROTEIN_PAIRS>
```

For performing the **Biological-and-Topological-External evaluation**:

..) Run the Topological-Feature extractor tool from `PPI_Challenge_submission_Becchetti_Fazzone_Martini/code/topological_feature_extractor/bin` in the following way:

```
    ..) $ java -Xms12g -Xmx12g algos.TopologicalFeaturesExtractor <i|e>
<directory_containing_the_input_file> <input_PPI_file_name>
<output_EXISTING_directory_that_will_contain_the_ENTIRE_TEST_set>
    ..) Example:
        $ java -Xms12g -Xmx12g algos.TopologicalFeaturesExtractor e
/Users/ikki/Dropbox/PPI/PPIChallenge/datasets/PPI_challenge_official_datasets
HuRI.csv
/Users/ikki/Dropbox/PPI/PPIChallenge/datasets/PPI_challenge_official_datasets/HuRI_
_TEST
```

Run the External-Validator tool from `PPI_Challenge_submission_Becchetti_Fazzone_Martini/code/ranker_and_evaluator` in the following way:

```
    ..) $ python3 biological_and_topological_external_validation.py
<directory_that_contains_the_ENTIRE_TEST_set>
<output_EXISTING_directory_that_will_contain_the_FIRST_500_PREDICTED_NON_INTERACTING
_PROTEIN_PAIRS>
```

COMPUTING ENVIRONMENT

All the programs were run on a MacBook Pro, macOS 10.15.7 with 16G of main memory and 2.9GHz Quad-Core Intel Core i7 processor.

EXTERNAL PACKAGES/LIBRARIES

The following python dependencies are required:

- Python == 3.8.5
- Java ==
- Sklearn == 0.23.1
- Pandas == 1.1.0
- Numpy == 1.19.1

- NetworkX == 2.4

ADDITIONAL DATASET USED IN THE METHODS

Protein sequences have been downloaded from UniprotKB (Knowledge Based) using their API [4].

METHOD DESCRIPTION

For predicting the interaction between two proteins we used the following three scores: **MaxSimScore**, **PAscore**, and **SeqScore**. The first two scores are based only on topological features of the PPI network, instead, the later is based on biological features of the single proteins.
In the following we describe all these three features together with the combination method we used to obtain a single final score.

MaxSimScore: Taking inspiration from Chen et al. [2] we designed the **MaxSimScore** in the following way:

$$MaxSim(u,v) = \max_{\beta\in\Gamma(v)} J(\Gamma(u),\Gamma(\beta)) + \max_{\alpha\in\Gamma(u)} J(\Gamma(\alpha),\Gamma(v))$$

Where Γ(x) is the set of neighbours of node x in the PPI network, J(Set_X, Set_Y) denotes the Jaccard similarity between Set_X and Set_Y.

PAscore (Preferential Attachment): As proposed by Kleinberg et al. [1], the **PAscore** is defined on a pair of nodes as the product of the degree of the two nodes.

Protein Sequence Interaction Score (SeqScore): This score is computed on a pair of not interacting proteins in the PPI network, according to the following algorithm (that is heavily inspired by the PIPE method described in Pitre et al. [3]):

```
.) SeqScore(a, b, PPI(V,E)):
..) return MAX(DirectedSeqScore(a, b, PPI(V,E)), DirectedSeqScore(b, a, PPI(V,E)))

.) DirectedSeqScore(a, b, PPI(V,E))
..) $S_a$ <-- GetProteinsWithSimilarPrimaryStructureTo(a)
..) $R_a = \{ v \in V \mid (v, s) \in E, s \in S_a\}$
..) $S_b$ <-- GetProteinsWithSimilarPrimaryStructureTo(b)
..) return $|R_a \cap S_b|$

.) GetProteinsWithSimilarPrimaryStructureTo(p)
..) return all proteins that have at least one sub-sequence of 20 amino acids in their Primary Structures in common with the Primary Structures of p.
```

Proposed Method with Only-Topological Features: This method simply sorts all the pairs of non interacting proteins in descending order of **MaxSimScore** first, and then in descending order of **PAscore** (**PAscore** is used as tie-breaking rule).

Proposed Method with Biological and Topological Features: This method simply sorts all the pairs of non interacting proteins in descending order of **MaxSimScore+SeqScore** (after normaliz them) first, and then in descending order of **PAscore** (**PAscore** is used as tie-breaking rule).

TIME COMPLEXITY ANALYSIS (optional)

Not reported.

References

[1] Jon Kleinberg, David Liben-Nowell, “**The Link-Prediction Problem for Social Networks**”, https://doi.org/10.1002/asi.20591

[2] Chen, Yu and Wang, Wei and Liu, Jiale and Feng, Jinping and Gong, Xinqi, “**Protein Interface Complementarity and Gene Duplication Improve Link Prediction of Protein-Protein Interaction Network**”, Frontiers in Genetics, Volume 11, Pages 291, Year 2020 https://doi.org/10.3389/fgene.2020.00291

[3] Pitre, Sylvain Dehne, Frank Chan, Albert Cheetham, Jim Duong, Alex Emili, Andrew Gebbia, Marinella Greenblatt, Jack Jessulat, Mathew Krogan, Nevan Luo, Xuemei Golshani, Ashkan, “**PIPE: a protein-protein interaction prediction engine based on the re-occurring short polypeptide sequences between known interacting protein pairs**”, BMC *Bioinformatics*, Volume 7, Issue 07, 27 March 2006, https://doi.org/10.1186/1471-2105-7-365

[4] Anne Morgat, Thierry Lombardot, Elisabeth Coudert, Kristian Axelsen, Teresa Batista Neto, Sebastien Gehant, Parit Bansal, Jerven Bolleman, Elisabeth Gasteiger, Edouard de Castro, Delphine Baratin, Monica Pozzato, Ioannis Xenarios, Sylvain Poux, Nicole Redaschi, Alan Bridge, “**The UniProt Consortium, Enzyme annotation in UniProtKB using Rhea**”, *Bioinformatics*, Volume 36, Issue 6, 15 March 2020, Pages 1896–1901, https://doi.org/10.1093/bioinformatics/btz817