Лабораториска вежба 4
Предвидување на врски

Податочно множество – Cora

Податочното множество Cora се состои од 2 708 научни публикации класифицирани во една од седум класи. Мрежата се состои од 5 429 врски. Секој јазол претставува научен труд, а рабовите помеѓу јазлите претставуваат цитирана врска помеѓу двата документи. Секоја публикација во множеството е опишана преку вектор со 0/1 вредности кој покажува отсутност/присутност на соодветниот збор од речникот во насловот на публикацијата. Речникот се состои од 1 433 уникатни зборови. Податочното множество може да се преземе од следниот линк. Датотеката README обезбедува повеќе детали.

Задачи

Задача 1 – Предвидување на врски со мерки за сличност (3 поени)

Во оваа задача треба да се имплементира предвидување на врски од графот кој произлегува од податочносто множество Cora. Вчитајте го графот од датотеката со име „cora.cites“, која што претставува запис на листата на врски од графот. Работете со најголемата поврзана компонента од дадениот граф. Од креираниот граф отстранете 10% од врските како позитивни примероци за тестирање. Дополнително, одберете исто толку врски во графот кои не постојат и ќе претставуваат негативни примероци при тестирањето.

Искористете ги Adamic-Adar (**`nx.adamic_adar_index`**), Jaccard Coefficient (**`nx.jaccard_coefficient`**) и Preferential Attachment (**`nx.preferential_attachment`**) за предвидување на врски во графот.

Со предвидените и постоечките врски да се пресмета:

- ROC AUC score
  - **`sklearn.metrics.roc_auc_score`**
- Average Precision
  - **`sklearn.metrics.average_precision_score`**

Задача 2 – Предвидување на врски со Random Walk (3 поени)

Да се извршат дадените барања од задача 1 преку предвидување на врски со алгоритмот Random Walk. За потребите на оваа лабораториска вежба дадена е имплементација на Random Walk with Restart во датотеката **`random_walk.py`**. Какви се добиените резултати?

Задача 3 – Предвидување на врски со автоенкодер (4 поени)

Да се извршат дадените барања од задача 1 со предвидување на линкови со автоенкодер. Овој автоенкодер е составен од енкодер кој ги мапира јазлите во латентен простор, по што следува декодер кој се обидува да ја реконструира матрицата на соседство од латентниот простор креиран со енкодерот. Дадена е имплементација **longae**, и за тренирање на моделот потребно е да ја извршите скриптата longae/train_reconstruction.py со следната команда:

```
python longae/train_reconstruction.py data/com-dblp.ungraph.txt 1
```

каде што **`data/com-dblp.ungraph.txt`** е локалната патека до графот, а **`1`** претставува CPU единица за тренирање на моделот (за GPU може да се користи 0).

Во скриптата **`train_reconstruction.py`** даден е пример за реконструкција на даден граф. Во оваа задача потребно е да го истренирате автоенкодерот, така што на влез на мрежата ќе влегува графот на кој му недостасуваат врските употребени како позитивни примероци за тестирање. Излезот ќе биде вистинската матрица на соседство во која не недостасуваат врски. По тренирањето на автоенкодерот зачувајте ги тежините и потоа направете предвидување на матрицата на соседство.

Со овие предвидувања извршете ги барањата за евалуација како во претходните задачи и направете споредба на новиот метод со претходните методи (време, перформанс, погодност, и сл.).