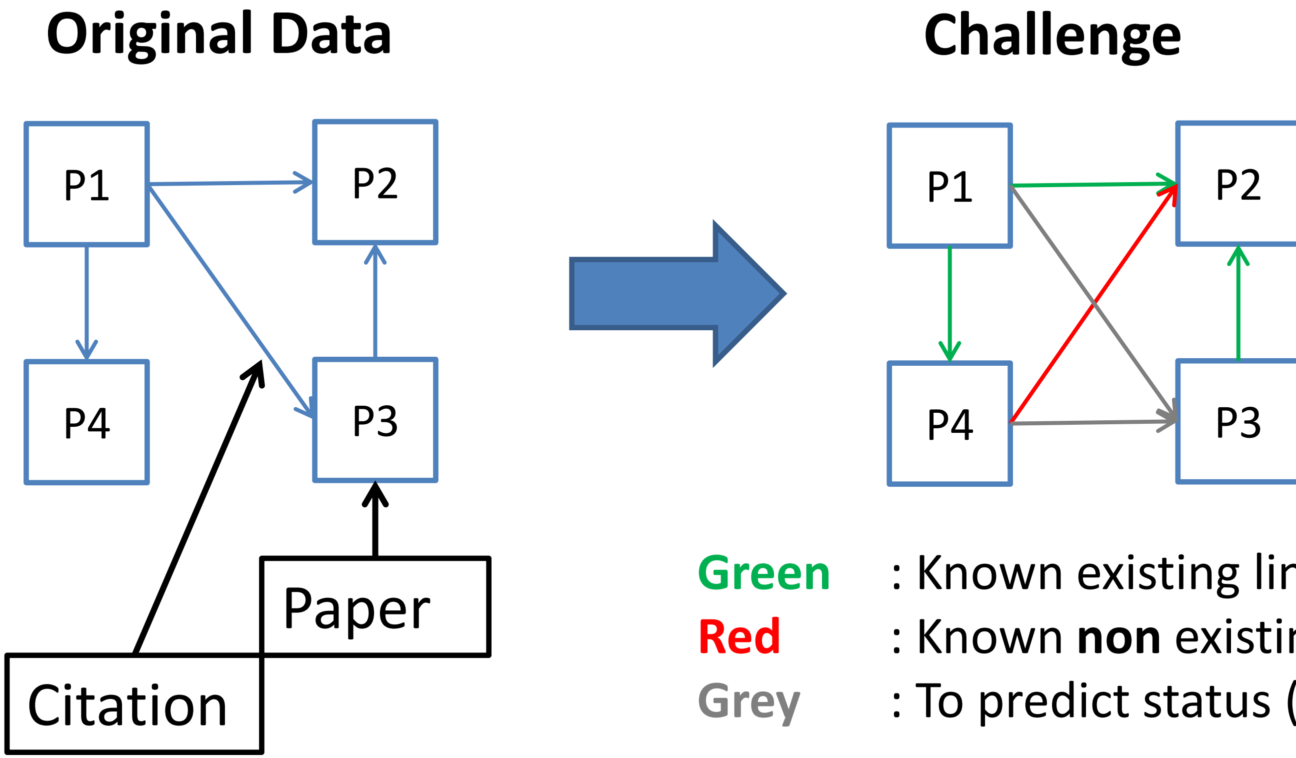
Link Prediction Data Challenge

Description and Key Points

The Task



Green : Known existing links (0)
Red : Known **non** existing links (1)
Grey : To predict status (0/1)

Besides the graph we also have information on the papers (title, authors, abstract)

Classification Task

- **training_set** <source,target,class> : pairs of papers and whether there is an edge between the two (1 means there is an edge between source and target)
- **node_information :** information on papers
- **testing_set** <source,target> : classify the pair of papers
- **example_simple_features** : example python code of the classification task

Example - Baseline

- Stem text : Porter Stemmer
- Simple similarities:
  - Title: overlapping terms
  - Authors: overlapping authors
  - Difference between publication years
- SVM classifier
  - Evaluate classifier on sample of data
  - k-fold validation

Evaluation

- F1 score: harmonic mean of precision and recall
  - $F1 = 2\frac{p*r}{p+r}$
  - Precision : How many did I get correct $p = \frac{tp}{tp+fp}$
  - Recall : How many of the desired class were correctly retrieved $\mathrm{r} = \frac{tp}{tp+fn}$

Improving the Baseline

- Text based :
  - Cleaning:
    - Data analysis: Are all terms in the text needed?
    - Part of speech tagging for removing terms
  - TF-IDF based features
  - Topic modeling
- Graph Based:
  - Node properties : degree, pagerank
  - Clustering coefficient
  - k-core number
- Similarity metrics:
  - Jaccard Index
  - Cosine similarity

TF-IDF vectorization

- Sklearn TfidfVectorizer

```
vectorizer = TfidfVectorizer(stop_words="english")
TFIDF_matrix = vectorizer.fit_transform(corpus)
```

- Corpus: a list of documents as strings
- TFIDF_matrix : Document X Terms (sparse) matrix
- Fit/transform:
  - Fit learns the dictionary terms and their importnacr
  - Transform computes TFIDF

TfidfVectorizer

- Parameters:
  – **Tokenizer :** Function to split strings into words. Override this if you want more than space based tokenization
  – **stop_words**: words to ignore
  – **ngram_range :** range of how many tokens should we use as “one term” (min,max)
  – **max/min_df : values in [0-1] range;** ignore words with frequency higher/lower than the specified
  – **vocabulary :** use only terms from this pre-computed list

Part of Speech Tagging

- Are all terms useful?

```
#assume tokens holds a set of words
tagged_tokens = nltk.pos_tag(tokens)
#tagged_tokens=[ (token, tag),…]
```

- (some) possible tags:
  - Verbs:VB,VBD,VBG,VBN,VBP,VBZ
  - Nouns: NN,NNP,NNPS,NNS
  - Adjectives: JJ,JJR,JJS
  - Other : ***nltk.help.upenn_tagset()***

NetworkX

```
import networkx as nx
```

Import library

```
G = nx.DiGraph()
```

Create a new directed graph

```
G.add_edges_from([("A","B"), ("C","A")])
```

Add nodes and edges edges

```
print G.in_degree()
print G.out_degree()
```

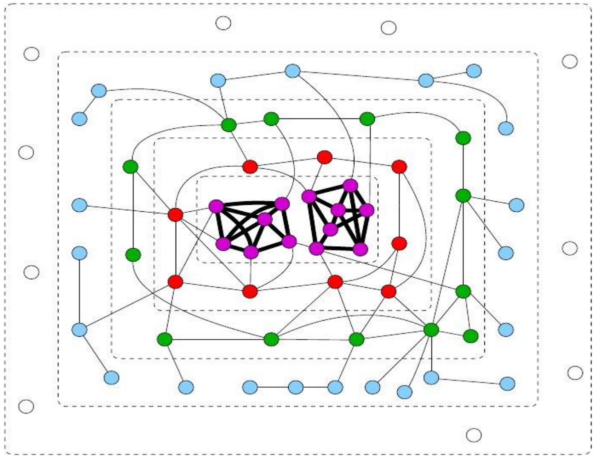
Print the in-degree and out-degree of the nodes

```
print G.neighbors("A")
print G.neighbors("B")
```

Print the neighborhood nodes of A and B

Graph Properties

- Clustering Coefficient: $c_u = \frac{2T(u)}{\deg(u)(\deg(u)-1)}$
  - T(u) : the number of triangles u belongs to
- k-core number:
  - k-core : the maximal graph where all nodes have at least k neighbors
  - k-core number of node: the maximum k for which a node belongs to a k-core



Latent Semantic Analysis (LSA)

- Assume matrix A with TF-IDF values

```
U, S, V = np.linalg.svd(A)
```

- Intuition: The collection is a produced by a collection of topics.
  - U: document to concept association. Each vector represents the topic weights for each document
  - S: A weight for each concept
  - V : term to concept association. Each vector represent the weight of a term to a topic
- We can project A directly to the concept space:

```
#keep only the most important topics
M = np.dot(A,V[:k,:].transpose())
```

Similarity Metrics

- Jaccard Index (sets A, B) :
  - $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$
  - Percentage of terms that overlap over possible terms
- Cosine similarity (vectors A, B):
  - $similarity = \frac{A \cdot B}{|A||B|}$
  - If we subtract the vector means then we get the Pearson Correlation

Further improvements

- Other classifiers
  - You can explore other classification algorithms from sklearn
- Optimizing hyper-parameters
  - Sklearn can automatically produce the scores of a classifier over a grid of possible values for the hyper-parameters (model_selection)

THE END

What else can you think?