M35216F: Research Paper Classification Challenge

Michalis Vazirgiannis and Giannis Nikolentzos

École Polytechnique, France

May 2018

1 Description of the Challenge

The goal of this challenge is to study and apply machine learning/data mining techniques to a research paper classification problem. Research paper (or article) classification is the task of assigning a research paper into a set of one or more predefined categories (i.e., classes). In our setting, these classes correspond to the journals in which the papers were published. Such techniques find applications in several domains, such as in managing collections of research articles. In this project, you will evaluate your methods on a dataset of research papers from the field of high energy physics. More specifically, you are given both a citation graph consisting of several thousands of research articles, and some other properties of these articles (e.g. their abstracts). The problem is very related to the well-studied problems of text categorization and node classification. The pipeline that is typically followed to deal with the problem is similar to the one applied in any classification problem; the goal is to learn the parameters of a classifier from a collection of training research articles (with known class information) and then to predict the class of unlabeled articles.

The challenge is hosted on Kaggle[1], a platform for predictive modelling on which companies, organizations and researchers post their data, and statisticians and data miners from all over the world compete to produce the best models. To participate in the challenge, you can use the following link: `https://www.kaggle.com/t/cb1776efe08e4d7bb080e1a7e96a4aef`.

2 Dataset Description

As mentioned above, you will evaluate your methods on a dataset of research papers from high energy physics. You are given the following files:

1. **Cit-Hep.txt**: a citation network from high energy physics stored as an edgelist. Nodes correspond to research articles and edges to citation relationships. For example, if a paper i cites paper j, the graph contains a directed edge from i to j. The graph consists of $27,770$ vertices and $352,807$ directed edges in total.

2. **node_information.csv**: for each research article out of the $27,770$ articles, this file contains the article's (1) unique ID, (2) publication year (between 1993 and 2003), (3) title, (4) list of authors, and (5) abstract.

3. **train.csv**: it contains $15,341$ labeled research articles. Each row of the file contains the ID of a research article and the journal in which it was published.

[1] `https://www.kaggle.com/`

Class	# of Train Articles
Phys.Lett.	2851
Nucl.Phys.	2836
Phys.Rev.	2555
JHEP	1547
Int.J.Mod.Phys.	747
Mod.Phys.Lett.	740
Class.Quant.Grav.	437
J.Phys.	437
J.Math.Phys.	418
Commun.Math.Phys.	317

Table 1: Description of the 10 largest classes

4. **test.csv**: this file contains the IDs of $3,836$ research articles. Each of these articles belongs to one of the 28 possible classes. The final evaluation of your methods will be done on these articles and the goal will be to predict the category to which each article belongs.

As regards the 28 classes, as mentioned above, they correspond to physics journals. The 10 largest classes are illustrated in Table 1. As you can see, the dataset is highly imbalanced.

3 Evaluation

The performance of your models will be assessed using the multi-class logarithmic loss measure. This metric is defined as the negative log-likelihood of the true class labels given a probabilistic classifier's predictions. Specifically, the the multi-class log loss is defined as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij})$$

where N is the number of samples (i.e., research articles), C is the number of classes (i.e., the 28 journals), y_{ij} is 1 if sample i belongs to class j and 0 otherwise, and p_{ij} is the predicted probability that sample i belongs to class j.

4 Provided Source Code

You are given two scripts written in Python that will help you get started with the challenge. The first script (`graph_baseline.py`) uses solely graph-based features with a logistic regression classifier for making predictions, and achieves a log loss score of 2.35 on the public leaderboard. The second script (`text_baseline.py`) uses features extracted from the abstracts of the reasearch articles along with a logistic regression classifier. This model achieves a log loss score of 2.47 on the public leaderboard. As part of this challenge, you are asked to write your own code and build your own models to predict the journal in which each research paper of the test set was published. You are advised to use both graph-theoretical and textual information.

5 Useful Python Libraries

In this section, we briefly discuss some tools that can be useful in the project and you are encouraged to use.

- A very powerful machine learning library in Python is `scikit-learn`[2]. It can be used in the preprocessing step (e.g., for feature selection) and in the classification task (a plethora of classification algorithms have been implemented in `scikit-learn`).
- Since you will deal with data represented as a graph, the use of a library for managing and analyzing graphs may be proven important. An example of such a library is the `NetworkX`[3] library of Python that will allow you to create, manipulate and study the structure and several other features of a graph.
- Since you will also deal with textual data, the Natural Language Toolkit (`NLTK`)[4] of Python can also be found useful.

6 Details about the Submission of the Project

Your final evaluation for the project will be based on your position on the leaderboard, on the log loss that will be achieved on the test set, as well as on your total approach to the problem. Specifically, final grades will take into account the following 3 components: (1) leaderboard score (40%), (2) submitted report (50%), and (3) code quality (10%). As part of the project, you have to submit the following:

- A 2-3 pages report, in which you should describe the approach and the methods that you used in the project. Since this is a real classification task, we are interested to know how you dealt with each part of the pipeline, e.g., how you created your representation, which features did you use, if you applied dimensionality reduction techniques, which classification algorithms did you use and why, the performance of your methods (log loss and training time), approaches that finally didn't work but is interesting to present them, and in general, whatever you think that is interesting to report.
- A directory with the code of your implementation.
- Create a `.zip` file containing the code and the report and submit it to the platform.
- **Deadline: 20/6/2018 23:59 (UTC+02:00)**

[2] `http://scikit-learn.org/`
[3] `http://networkx.github.io/`
[4] `http://www.nltk.org/`