

Five Hundred Deep Learning Papers, Graphviz and Python

Daniele E. Ciriello

Abstract. I invested days creating a graph with PyGraphviz to represent the evolutionary process of deep learning's state of the art for the last twenty-five years. This is the final result:

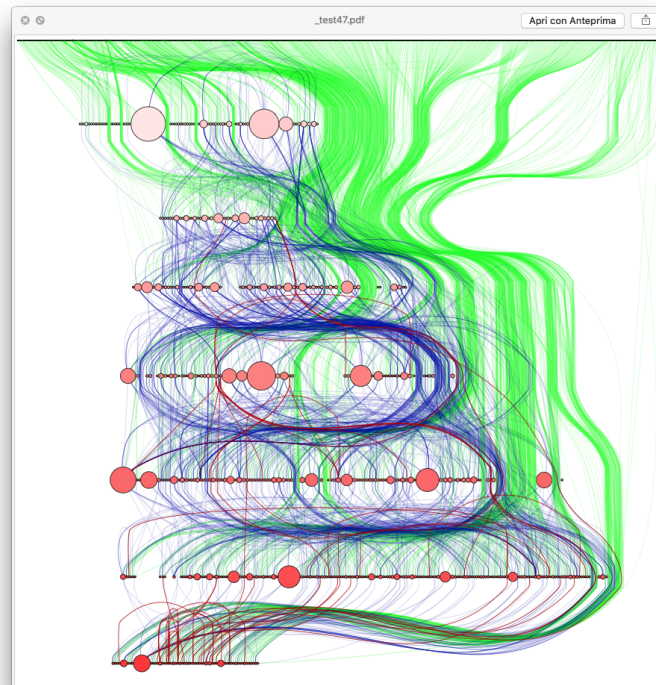


Fig. 1. Graph Test 0

Keywords: deep learning, graph, graphviz, python

1 Introduction

I am currently writing my computer engineering master thesis about Deep Learning. When I came to the state-of-art part I thought I could build up something interesting by making a graph of the most relevant papers for the last twenty-five years, the only problem is that I don't have much experience with GraphViz so after I built a sort of JSON archive by scraping (sorry for that) Google Scholar, I made a lot of tries to figure out how GraphViz works and how I could visualize the results in a meaningful way. In this post I will show how in brief how I obtained the papers data and how I came to the final result above.

2 Obtaining Data

The first thing I was needing was the papers data, I wanted to create a graph in which each node represents an author or a paper, each article is connected with their authors through 'Author->Paper' arches and each paper is connected with the papers who cited them through 'Paper->Paper' arches. I also thought about sorting the papers according to the date of publication.

In other words I needed a list of relevant papers and for each paper:

- Title
- Authors
- Publication date
- Bibliography (or a "cited by" list)

So I searched on Google and I found out that arxiv.org has an API that allows programmatic access all of the e-prints hosted by them, "good!" I thought, but when I went through the arXiv API User's Manual I found out that it's impossible (correct me if I'm wrong), to get information about the bibliography of the papers or to search all the papers "cited by" another paper, the only hosting service that give this piece of information I found was Google Scholar, plus it gives the possibility to search all the papers citing a specific paper by the 'cites' parameter, so I searched "scholar api" on Google and what I saw is this:

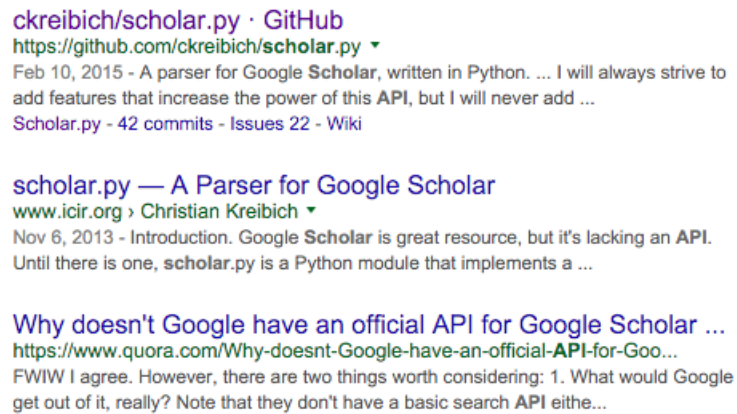


Fig. 2. “scholar api” results

After I read some reasons about why Big G doesn’t offer an API service I went on github and tried `scholar.py{:target="_blank"}`, but it outputs the results only in TXT or CSV file formats, so I forked`{:target="_blank"}` it and added some feature like exporting the results in a JSON file, setting the starting result number by adding the ‘start’ parameter in the search query and the ability to get as much results as one wants (results seems to be limited by circa 1000 on server side though), so I started getting the “deep learning” results saved on a list of dictionaries in a JSON file with this schema:

```
[
  {
    "num_versions": 3,
    "url_citation": null,
    "title": "Deep self-taught learning for handwritten ...",
    "url": "http://arxiv.org/abs/1009.3589",
    "url_versions": "http://scholar.google.com/scholar?...",
    "authors": "F Bastien, Y Bengio, A Bergeron\u2026",
    "excerpt": "... 1 Introduction Deep Learning has...",
    "url_PDF": null,
    "num_citations": 4,
    "cluster_id": "13430136242568199344",
    "year": "2010",
    "url_citations": "http://scholar.google.com/scholar?..."
  },
  ...
]
```

As you can see there are obvious limitations to this approach but I will come to them later, because the biggest obvious problem is that after a few dozen of results the command line script stops working because of Google's reCaptcha page that obviously means that Google knows you are doing something not conforming to their TOS. What to do then?

I could not wait hours nor change my public IP every time for sure, so given my little experience with Selenium, I implemented what I call a "supervised scraper" starting from my scholar.py fork and published it as PyScholar (Is still a work in progress, and I haven't created yet an installation guide but to run it you simply have to install selenium and eventually other needed packages through pip, you also need the selenium correspondent stable version of Firefox installed on your machine).




The main feature of PyScholar is that it instantiate a Firefox window, and I know it is boring for many people but this permit users to see when the script pauses because of a captcha, put the solution and let the script continues typing 'continue' or 'c' in the terminal window, at the moment it makes use of pdb to pause the script. Please note that PyScholar is only a prototype, there would be various way to improve it (and you can contribute if you want). Anyway I wanted to make use of selenium webdriver for two main reasons: the first one is that I wanted to see what was going on while scraping, secondly but not certainly for importance, the new "I'm not a robot" reCAPTCHA:

I also tried to minimize the number of times this really annoying challenge shows up, by saving all Firefox profile files and load them each time the tool stops and starts and I have to say that with this trick the "I'm not a robot" page shown up to me very few times during this job, let 'say that from my experience on 100 requests a standard (automatically generated) captcha is shown every 25 pages circa, while the "I'm not a robot" is shown every 500 requests circa (from what I remember). The funny thing is that Google uses reCaptcha service to build their machine learning training datasets, among other obvious uses.

So what PyScholar did was to search for the most relevant 'deep learning' papers in Scholar, excluding pure citations and patents, and saving the results in a JSON file as said before, so I had the "first layer" of papers, then I added a feature on PyScholar to input a JSON file and getting at most the first X relevant papers citing A for each paper A in the first layer of papers.

I thought I could get a relevant representation of the deep learning state of art evolution for the last years by skipping all results with less than 10 citations and applying $X = 50$ to build a second JSON list, the "second layer" of papers, and searching again for the $X = 50$ most relevant papers which cite B, for each paper B in the second layer's papers excluding the B papers that with less than 10 citations (I also decided to keep in the second layer B all the papers published in the years 2014 and 2015). Thus I created the third and last papers layer, after that I figured out I could stop because practically all of the papers citing the papers in the third layer (excluding the papers not published in the last two years with less than 10 citations) already resided in one of the layers.

Select all images with **steak**.

Report a problem

[Verify](#)

Fig. 3. Python is not a robot

3 Results Limitations

As I said before if you see the JSON output example, you can see there are some limitations to this approach, the first big one is that in the author field there aren't only authors, there we have the publication year (that scholar.py extracts and put at the 'year' key), the research field or other information (like the "arXiv preprint arXiv" in the example) that I honestly dropped because some result contains too much authors that the string get truncated, and other characters I had to remove, so I took all authors I could from this field, conscious that I lost some authors for sure. Rarely also the extraction of the publication year from the 'authors' value gave me problems, as we will see below.

4 GraphViz + Python = Let the fun begin

After getting enough data to build a meaningful graph I started searching the best way to build and show a big graph, preferably using Python, suddenly I found GraphViz. I used it before implicitly trough Doxygen and remembered that it was a very powerful tool, so I installed PyGraphViz, downloaded the Dot documentation and started playing around with the graph creation process, putting all author nodes on a row and all paper nodes below the authors, connecting the authors to their correspondent papers and the papers to correspondent papers which cited them. After some tries I get the first meaningful result, limiting the minimum number of citations and the number of arches in the graph:

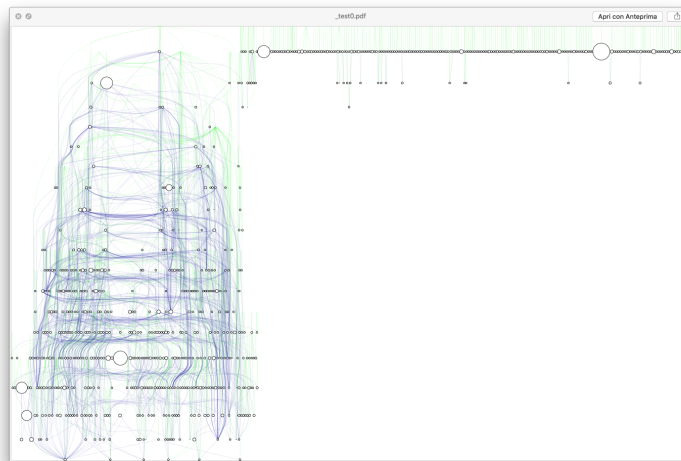
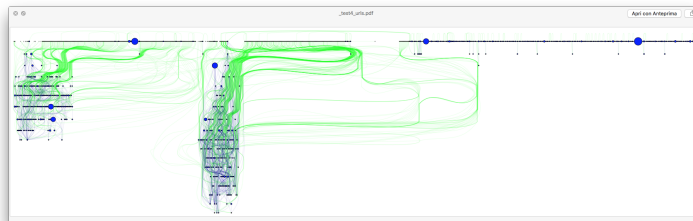
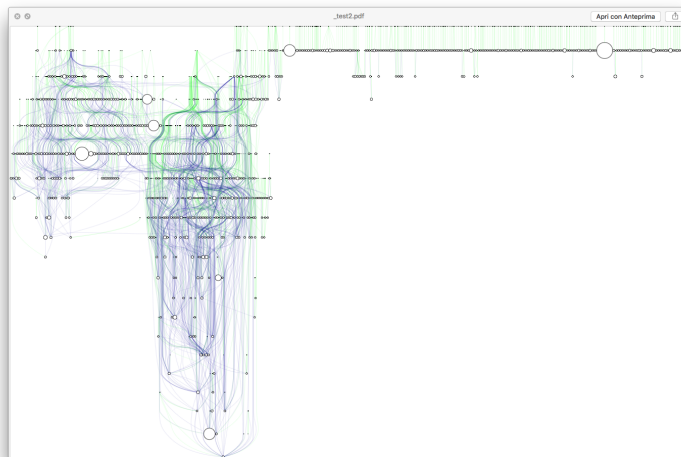


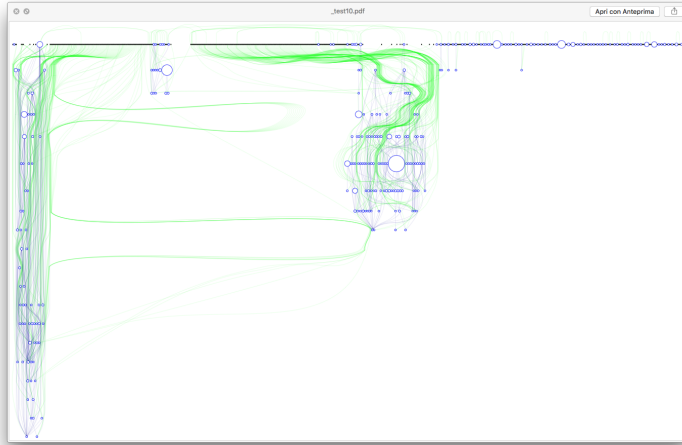
Fig. 4. Graph Test 0

Note that to create the PNGs I had to take screenshots of QuickView because I tried to export the PDF in various ways but waiting times are too much long because the process is computationally very hard (or more likely I am doing something wrong).

I coloured green the paths representing a “wrote” relation (authors are almost invisibles here though), blue the paths represent a “cited_by” relation and then I sized the paper nodes depending on the total number of citations received. Obviously there still were a lot of bugs to fix: sizes weren’t meaningful at all, position parameters were ignored, node colouring to be implemented, etc. But I think it was an exciting starting point.

So I started playing with both the Dot parameters and various parameters created by me to select the papers to be shown, then I added the paper’s links to the paper nodes:

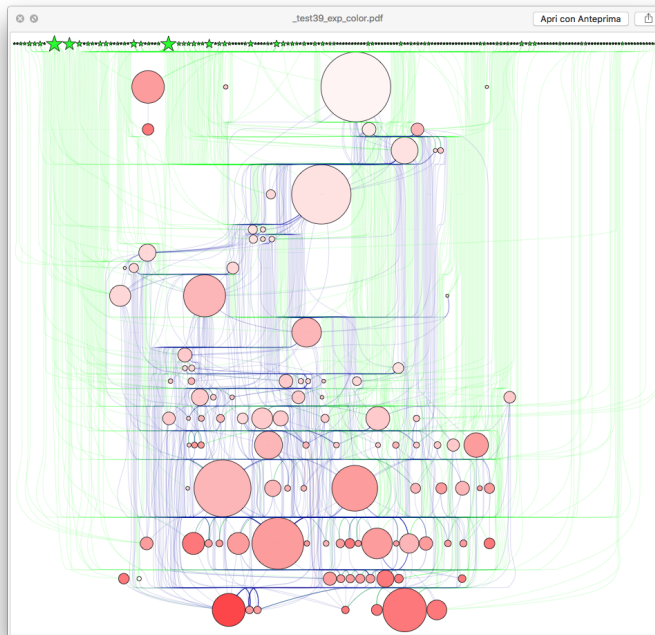




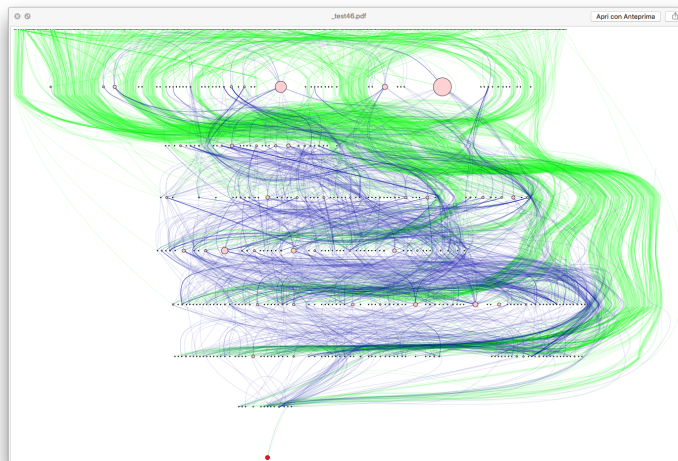
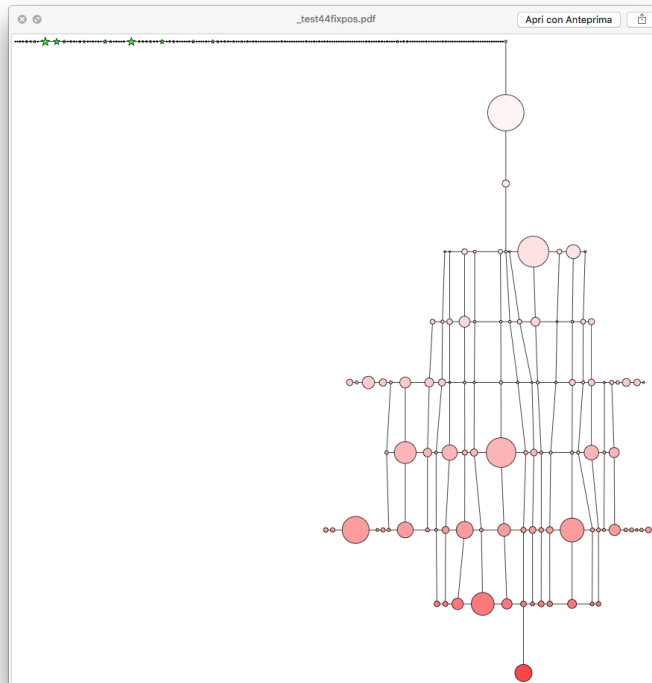
As perhaps you have noticed, I started understanding that there are two completely uncorrelated group of papers in many tests, by clicking on the nodes on the left I got it, there are a lot of useless articles, because in the results there were a lot of papers talking about deep learning, in psychology terms: how to teach to students in a proper way, various studies regarding students' learning processes, papers totally unrelated to neural networks or machine learning, my fault, I could spend more time refining the search query. By the way I figured out a way to remove the authors and their relative papers from the dataset, when the uncorrelation became evident in the graph, by fixing all author nodes on a row in the graph so I could select all authors of the uncorrelated papers, copy, paste in a file and remove them with minimum effort.

Then I implemented a paper nodes' sizing by "citations ratio" (citations / year) and started colouring the paper nodes depending on the publication year, also we can finally see the authors, in a green star shape, and finally sized depending on the number of written papers:





After removing some other outliers and some not-papers nodes like books or slides, I fixed the node colouring and positioning by adding invisibles paths through the nodes:



So I found the last outlier, a paper whose year was interpreted as 2080 instead of 2014 because of the limitations written before, finally I came to this graph:

Not-green paths are the citations starting from a paper A and ending to a paper B citing A, blue paths start from the first layer papers, red paths from the second and pink paths from the third one.

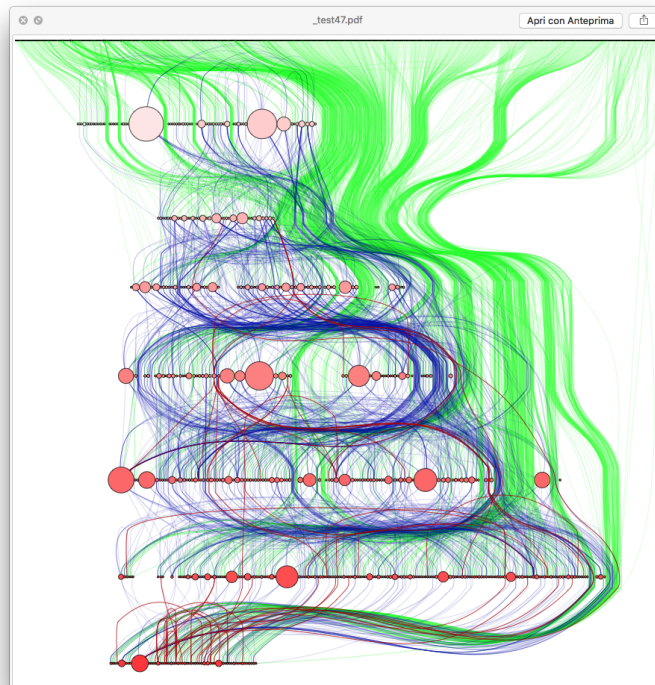


Fig. 5. Graph Test 0

I started with 1071 papers (probably most relevant papers citing papers in the first layer were already in the first layer) and ended with 513 unique relevant papers by 945 unique authors and a total of 1548 citations. I don't think this graph is complete, if I had more time I would enlarged the paper set by including papers inherent to at least "artificial neural networks" and something about "machine learning" but I have to pay bills and I cannot invest much time in this spin-off project at the moment. I would also appreciate any suggestions/critics.

5 Other Resources

You can find here a list of all the screenshots of the tests I kept. I also uploaded the final SVG file with a paper information popup when hovering nodes and correspondent PDF: `test48.svg{:target="_blank"}` and `test48.pdf{:target="_blank"}`. Keep in mind that there are a lot of things that could be improved: citations information when hovering the paths, more informations about the authors, I would also like to add some CSS effects to the SVG.