Julian Gonzalez

AI Final Project: PageRank Algorithm

I will be focusing on the PageRank Algorithm as the topic that I will write about. PageRank Algorithm seemed interesting to me after going over the Natural Language Processing chapter of our textbook. Chapter 22 Natural Language Processing overall caught my eye as it captures the interactions that our human language has with computers, artificial intelligence, and how all of the natural language information is acquired and examined by natural language algorithms. PageRank's original idea was first created in 1997 by Google. Google wanted to create their own way of searching the world wide web that was different from other search engines at the time.

The basic concept of the PageRank Algorithm was to remedy the problem that TF scores had with the way it ranked different websites when searching for them. PageRank algorithm is to make sure that search terms match with what someone is looking for, an example of this is looking up UTEP on google. Typically we expect UTEP's webpage to be at the top or among the top of the search results, this is what the PageRank algorithm is made to do. PageRank uses different criteria to rank the relevant search term with its website. First it checks for websites that mention the term "UTEP" and links to the page more than other websites and uses this to help place the relevant website at the top of the search results. One problem that arises from this is websites that spam the search term and have many links to that site page. What PageRank does is to weigh the links of higher quality websites more favorably. Higher quality website "status" is determined by being linked to by other high quality websites which as described in the textbook "The (PageRank) definition is recursive". This recursiveness is

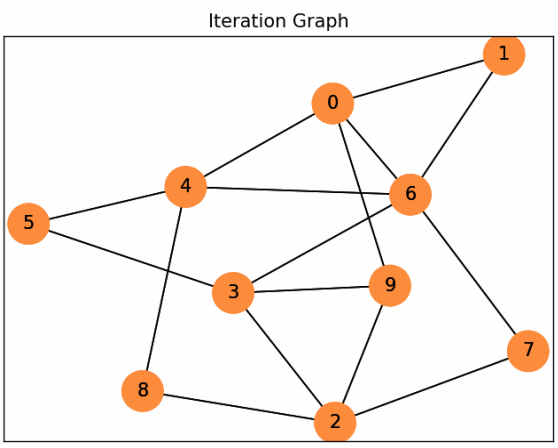
“bottomed out” by the way PageRank is given for webpages. Below is the PageRank of a page as described in the book:

$$PR(p) = \frac{1-d}{N} + d\sum_{i}\frac{PR(in_i)}{C(in_i)}$$

Where p is the page, PR is the PageRank, N is the number of page collection of websites, in_i being the web pages that link to the original (p) website and $C(in_i)$ being the number of counted “out links” or pages that link other websites on in_i pages. The damping factor or d is a constant that is determined by the random surfer model (which is assumed to be .85 in the textbook). It is the probability that a person or surfer in this case clicks on a link of a webpage. So overall PageRank of a certain website is the probability that a person will be on the given page at any time. PageRank can also be done iteratively with just having the pages with a value of 1, which can be iterated and updated until convergence is met.

For my implementation of the PageRank algorithm I opted for the iterative version of the algorithm. To implement an example of this problem I had used the networkx library for python to create a random graph that would be used on the algorithm. The way I had generated the code with networkx’s gnp_random_graph function would randomly generate a graph for me to use with my chosen amount of nodes and frequency of edges for each of the nodes. This would sometimes result in nodes with zero edges which would leave my program hanging when trying to generate the diagram of it. I was unable to solve this but instead opted to leave it but raise the probability that an edge would generate. Next was creating the PageRank function which was not too difficult given the formula above but required me to change the way

pages (p) where handled as they all need to be initialized to 1, done by using np.ones() function. Given the networkx graph that was generated(G) I gathered the Nodes from G and used it to create an adjacency matrix. With that matrix we can determine the the out degree value which would be used to find the weight (matrix divided by out_degree) The next part was easily done I created a for loop that would iterate and find PR given the PageRank formula until a max limit was reached or convergence was achieved with the given tolerance of 1e-6 which was compared to the error which was computed by getting the absolute value of the current PR value subtracted by the previous PR value. Once convergence was met I returned the PR which was then used for graphing with the matplotlib python library. An example of an output is shown below. This was the most time consuming part of the code but it does not directly pertain to the algorithm itself though I had many problems with generating the graph but eventually had solved them.



With the generated graph and corresponding gif you can see the way PageRank starts to give nodes with more edges a darker orange color. This corresponds with the way the algorithm gives pages with more inlinks higher priority through many iterations in this case.

Julian Gonzalez

Research Paper

Here I will be discussing the research paper that was related to my topic PageRank. The paper **Distributed PageRank Computation: An Improved Theoretical Study** by Siqiang Luo. This Piece was published with the AAAI conference in 2019. The paper explores the short falls that PageRank has, specifically the computational power that is needed to traverse large graphs given the distributed computation model. This model currently take big $O(\sqrt{logn})$ to find the PageRank of a given value. PageRank is used by many scientific communities involved in web algorithms, data mining and distributed networks, it is even still used by google today for some part of its search algorithm internally. So being able to improve the time complexity is still very beneficial today.

The paper proposes an improved time complexity of $O(log\sqrt{logn})$ to be able to calculate the PageRank among other improvements. Using their BPPR or Batch One-Hop Personalized PageRank which is based on an extension of PageRank called Personalized PageRank they are able to use their compute PageRank (algorithm 4) they are able to achieve the improved time complexity of PageRank. The paper also goes into detail about advancements in other aspects of PageRank such as the communication size down to $Ologn$ and its edge bandwidth $O(log^{3}n)$. I can see how this improvement in PageRank will allow further work and even better improvements into an algorithm that is still used by many, even its own creators. Allowing for its modification to be used for many different applications.

References

1. S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Upper Saddle River: Prentice-Hall, 2010.
2. S. Luo, “Distributed PageRank Computation: An Improved Theoretical Study,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4496–4503, 2019.