Customized Random Walk for Generating Wikipedia Article Recommendations

Jocelyn Hickcox and Chris Min

December 8, 2015

Introduction

Suggesting content for online users based on their previous activity is an ongoing problem for websites. Personalized content recommendations are influential in deciding customer satisfaction and make it easier for users to focus on only the information that is relevant to their interests. Indeed, one way to look at personalized recommendations is as a clustering problem: good recommendations will be the items that are most similar to the items in the user's history. In this way, generating recommendations takes on the second motivation of gauging what it means for an entity to be similar to a set of entities.

In this paper, we look at a network of Wikipedia articles and evaluate different methods for generating recommended articles based on a set of starting articles. As Wikipedia is not generally a site that needs to provide personalized recommendations, our primary motivation here is to determine how well content similarity in articles is captured by these methods.

There are two types of algorithms traditionally used in recommender systems, both of which have weaknesses when it comes to large networks. The first, collaborative filtering, compares a user's preferences with a large number of other users' preferences to return things they are likely to enjoy. This method, however, requires storing preferences for each user and keeping them up to date as they react to new things. This is clearly not ideal for first-time users and can be very memory-intensive as the number of users increases and many updates must be made. The other, content-based method looks instead at the items that the user has in their history and compares them to other items in the database and returns the most similar ones. Although this method doesn't require much more storage than is already required by having the options available, it does require complex and imperfect calculations of similarity between multifaceted items.

We instead look at a network-based approach to finding related content. Each node in the network represents a Wikipedia article, with directed links going from one article to another if there is a hyperlink within the first article to the second. Given a set of starting nodes, can we traverse the graph from each of these nodes and find the articles that are most related to the starting nodes? We first assess an algorithm used by Demovic et al. to address a similar problem within a network of movies, and then describe and analyze an algorithm we developed based on random walking to accomplish this task with a greater amount of control.

Related Work

The [2] (Demovic et al.) paper was our main inspiration for creating a network-based recommendation system. In this paper, they develop and rank four different algorithms for generating movie recommendations based on a set of starting nodes. Of the four algorithms, two were judged to be best according to qualitative rankings done by 30 experiment subjects. Of these two algorithms, one involves spreading activation to neighboring nodes based on the energy of its parent nodes and is very expensive to compute while not providing much more of an advantage over the other, faster algorithm. This faster algorithm is called the Mixing Colors Algorithm and is the one that we focused on the most in the paper.

The Mixing Colors Algorithm works as follows:

1. Mark each initial node as visited by a different color
2. Perform a simultaneous breadth-first search (BFS) on each initial node, coloring each visited node with the color of the node that visited it.
3. Once the desired number of nodes is colored by every color, return those nodes.

We implemented this algorithm on the Wikipedia network as a baseline method. This algorithm returns the set of nodes that are closest to the initial set of nodes. Since each round of the BFS is done simultaneously, the recommended nodes added at each round are the same number of hops from the initial nodes that color them. If the most relevant articles are just those that are the fewest hops away from the initial articles, then this algorithm should return the best results.

The [1] (Avrachenkov) paper outlines the strategy for the Monte Carlo random walk method that we used in our random walk search. The authors describe an alternative to the power iteration (PI) method of calculating PageRank: a Monte Carlo approach that incorporates information about previously visited pages. They argue that in many cases, even a single iteration of the algorithm is sufficient to rank the most relatively important pages. Their approach's other advantages include a naturally parallel implementation and the ability to continually update the PageRank as new nodes and edges are formed.

In [5], Richardson and Domingos propose a modification to the traditional PageRank algorithm by following links from one page to another with probability proportional to their relevance. By preprocessing the relevance between each page, they decrease the computational complexity of each query. They find that their model increases the quality of PageRank's results while remaining within reasonable range of modern time and storage standards. We apply their method of following links with probability proportional to their relevance to our random walk algorithms, using a variety of different edge weights to proxy for relevance.

[4 Ch 6.] (Manning) describes tf-idf weighting for information retrieval, a weighting scheme also useful for comparing the content of two bodies of text. Each document in the series is represented as a vector indicating the distribution of words that occur in the text. The ith value in a document vector represents the number of times that the ith word in the vocabulary occurs in that document, with the indexing consistent across all document vectors (i.e. the ith element refers to the same word in each vector). The document vectors are normalized so as not to bias towards documents with a greater number of words. This is the tf (term-frequency) component of tf-idf.

However, very common words like "the" or "a" would have large values in the document vectors and would influence the similarity of two document vectors despite their having little to do with the actual content of the document. To counteract this, we incorporate the inverse document frequency (idf), which punishes words that appear in a large proportion of the documents. Very common words will appear in most documents and so will be down-weighted in the tf-idf vector. Inverse document frequency of a term t is defined as:

$$idf_t = log(\frac{N}{df_t})$$

Where N is total number of documents, and df is the number of documents that the term t appears in. Multiplying tf * idf gives us our vector representation for each document. Because our documents are all represented by vectors in |V| - dimensional space, where |V| is the size of the vocabulary, we can determine the similarity of two vectors by taking the cosine of the angle between the two vectors. Identical vectors will have a cosine similarity of 1, whereas orthogonal (completely dissimilar) vectors will have a cosine similarity of 0.

In order to assess the quality of recommended articles, we need to have some way of judging how relevant the article is to the initial articles. To do this, we use the tf-idf vector representation of the text of each article and average the cosine similarities of each of the initial articles to the recommended node.

Model, Algorithm, Method

We used the Wikispeedia navigation paths network from Jure Leskovec[1]. This network contains around 4,500 Wikipedia articles as the nodes and about 120,000 directed links between them according to whether one node links to the other. Although we do not make use of the Wikispeedia path data, we used this dataset rather than the general Wikipedia dataset because it is smaller and more manageable than the entire Wikipedia network and because it contains the articles' text in an easy to parse format.

We generated a series of starting pages from which to find recommendations. We used a set of randomly generated starting article pairs to establish

[1]http://snap.stanford.edu/data/wikispeedia.html

our overview statistics with which we compare our different algorithms. We also hand-picked a variety of starting node pairs to see how our algorithms fared when applied to nodes with a different relationships. We included a set of neighboring nodes ('Jerusalem' and 'Israel'), as well as nodes with the longest shortest path between them ('Scheme_programming_language' and 'Black_panther' with a shortest path distance of 9). We also used two similar people ('Paul_McCartney' and 'Bruce_Springsteen'), and things within the same domain ('Boston_Terrier' and 'Rabbit'; 'Jazz', 'Saxophone', and 'Music').

We implemented the *Mixed Colors Algorithm* as described above to provide a baseline of similarity based on closest nodes to the starting set.

Our random walk algorithm takes in a graph, a set of starting article IDs, the number of steps the random walk can take, a restart probability of epsilon, and a method for weighting links. We ran our random walk with 250 steps, which tended to return between 3 and 35 recommendations depending on the weighting scheme.

A random walk was started from each initial node and proceeded for the allotted number of steps, and the nodes visited in each walk was recorded. At the end of all the random walks, the articles that were in every random walk path were returned as recommended articles.

When deciding which outlink from an article to follow during the random walk, the default action was to choose randomly from the available outlinks. Not all hyperlinks are created equally, however, and we were curious if we could leverage what we knew about the network to bias the random walk towards articles that are more or less relevant. We experimented with five weighting schemes in total, with the lack of a weighting scheme (i.e. randomly choosing between outlinks) making up the sixth. They are as follows:

WEIGHTING BY IN-DEGREE: Weight the outgoing links according to the in-degree of the destination node. The weight of each out-link is set to be the in-degree of the destination node, which we then normalize and treat as a probability distribution. Nodes with high in-degree tend to be more broad than nodes with low in-degree (for example, 'United_States' has very high in-degree whereas "List_of_areas_in_the_National_Park_System_of_the_United_States" has a lower in-degree. We expect that this weighting scheme will bias towards recommendations that conceptually overarch the initial nodes.

WEIGHTING BY INVERSE IN-DEGREE: While biasing towards higher-degree nodes may give you broader results, they may not be as specific to the initial nodes as one may wish. This weighting scheme again assigns a weight to each out-link according to the number of in-links to the destination node, but this time the weight assigned is the inverse of the in-degree. We expect that this weighting scheme will return fewer but more specific results.

WEIGHTING BY OUT-DEGREE and WEIGHTING BY INVERSE OUT-DEGREE: These weighting schemes are identical to the two above, except rather than weighting by the in-degree of the destination node we weight by that node's out-degree. The idea here is that nodes with high out-degree may cover a broader range of topics than nodes with low out-degree, which we suspect will be more confined. Out-degree may also correlate with article length, since a longer article has more opportunity for hyperlinks.

Weighting by PageRank: This weights each outlink according to the PageRank of the destination node. More "important" nodes are more likely to connect the random walk sets with other sets, so we expect more results to be returned with this weighting scheme. We do not expect that biasing towards pages with higher PageRank will return better recommendations, however. The nodes with the highest PageRank in Wikipedia tend to be too broad to be considered the intersection of two random nodes–for instance, 'United_States' and lists of things tend to have high PageRank.

Results and Findings

We ran each of our algorithms on 100 randomly selected pairs of starting articles, initially setting epsilon to 0.5 for the random walk algorithms, where epsilon is the probability that the random walker restarts at the initial node at a given time step. The results are show in in Figure 1 below. We see that the in-degree random walk performs best, while the out-degree and PageRank random walk algorithms come a close joint second. We may attribute the success of in-degree to the fact that articles with high in-degree may serve as pseudo-"reference articles" that are broader in content. Within the locality of the article in question, we may even conceptualize the graph as a hierarchical tree, where conceptually broader articles have many child nodes covering more specific topics, with the children referencing their parent through directed links. We may apply similar reasoning to the out-degree algorithm, assuming that longer, broader articles have outlinks to articles that dive deeper into specific topics. PageRank in many ways is founded upon the notion that both high in- and out-degree imply significance of a node, so we should expect performance similar to that of the two algorithms that use the two properties independently.

Given the success of both the in- and out-degree weightings, the poor performance of their inverses makes sense. The unweighted random walk and mixed colors algorithms' performances come in at the middle of the pack, as they do not weight their edges at all.

While our scoring methodology serves as a solid heuristic for the actual relatedness of articles, we believe it cannot outperform a human. With more time and resources, we would have liked to experiment with real human beings, asking them to evaluate the quality of recommendations put out by our various algorithms. In such a case, attributing a numeric score for relatedness between articles may become difficult. We could instead ask participants to drop unrelated articles from the set of recommended articles, thus evaluating recommendation sets based on the percentage of their articles that survived the participants' culling.

We found that 250 was a good number of steps to run the random walk algorithms for, as it returned a reasonable number of recommendations. This left ϵ, the restart probability, to optimize for. We used $\epsilon = 0.5$ for all previous experiments since the particular epsilon did not matter as long as it was

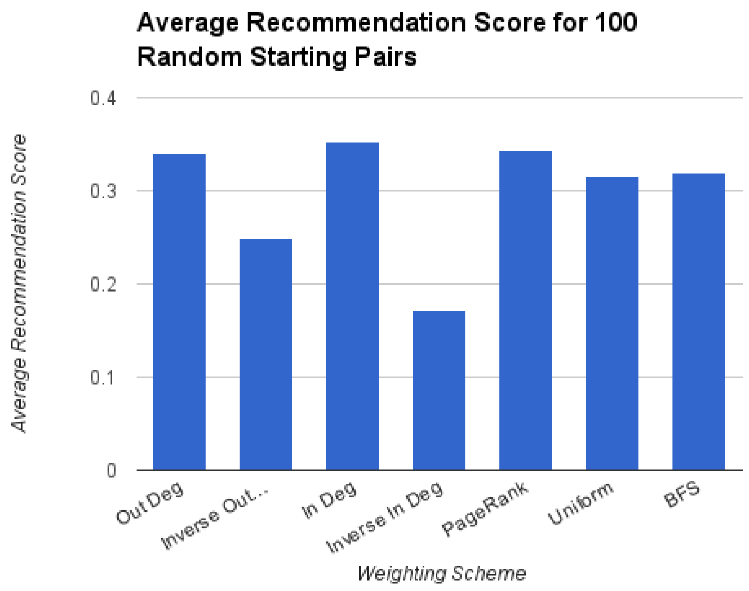

Figure 1: Average recommendation score from 100 random starting pairs for each recommendation method. Cosine similarities between a recommendation and each of the initial nodes are averaged across all samples. The same 100 starting pairs were used for each method. Labels correspond to the weighting methods referenced above, with "Uniform" meaning uniform probability of following any outgoing link, and "BFS" referring to the Mixed Color Algorithm.

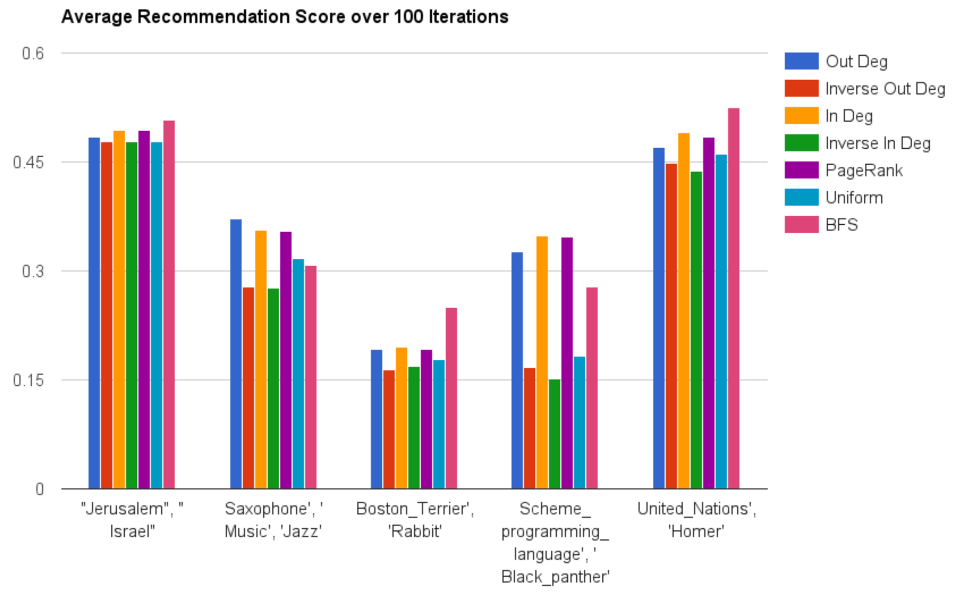

Figure 2: Average recommendation scores from different recommendation systems from different initial nodes. The starting nodes were selected to give a variety of situations. “Jerusalem” and “Israel” are both one hop away from each other and tend to have higher than usual recommendation scores. “Scheme_programming_language” and “Black_panther” are the farthest nodes apart in the network, with the shortest path from Scheme to Black_Panther 9 hops. The results for this seed has much greater variation than for other seeds. “Boston_Terrier” and “Rabbit” did not fare as well as expected, but the shortest path from Rabbit to Boston_Terrier is 4 hops, which is on the higher side for this network. There is a range of results for individual recommendation methods, depending on the nature of the starting nodes.

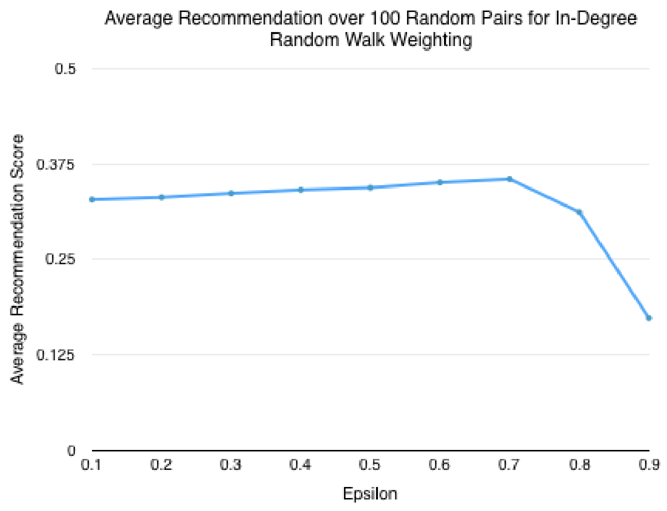

Figure 3: Average recommendation score for the in-degree weighted random walk system with different values of epsilon, where epsilon is the probability that the random walker returns to the starting node at a given time step. Results are averaged from 100 randomly selected initial node pairs within the network. The same 100 pairs were used for each value of epsilon. The average score climbs slowly but steadily until it reaches epsilon = 0.7, after which it plummets sharply. epsilon = 0 and epsilon = 1 were not included, since epsilon = 0 mimics a single depth-first search and epsilon = 0 will remain at the initial node forever.

consistent across all examples and our random walk algorithms were already performing better than the Mixed Color Algorithm in many cases. Taking the random walk algorithm with in-degree weighting scheme–our best performing algorithm–we averaged the recommended articles' scores for 100 randomly chosen starting node pairs for each different value of epsilon (Figure 3). We found that the the average recommendation score was highest for $\epsilon = 0.7$, but that the score dropped off steeply for values of epsilon greater than that. The recommendation scores for values of epsilon less than 0.7 were comparable to those at epsilon = 0.7.

High values of epsilon will make the random walker reset to the initial node quite often, preventing it from moving very far through the graph. Because the random walker only takes a set number of steps (250 in this case), it is less likely that the random walks will overlap, and most likely fewer recommendations will be returned. The question remains as to why these returned recommendations are not high scoring. One possibility is that beyond a certain value of epsilon, the exploration of the random walks are so limited that it can only generate a few recommendations of nodes with likely high in-degree. These hub nodes may not be very similar to the initial nodes, but serve as the only recommendations due to these limitations. There may be another explanation for this, however, which could be explored in future work.

Questions of recommendation quality aside, we also note the differences in computational complexity between the different recommendation algorithms. Assuming that the graph and its edge weights are preprocessed, the random walk algorithm runs in $O(kT)$, where k is the number of starting articles and T is the number of steps each random walker takes. On the other hand, the mixed colors algorithm runs in $\sum_{a-1}^{k} \sum_{t=0}^{T} n_{at}$, where n_{at} = the number of nodes at that are t hops away from the starting article. n_{at} grows exponentially, which can become problematic in runtime-sensitive situations. Particularly, given two initial pages, the algorithm must run for enough rounds so that two directed paths beginning from either of the initial pages may meet. With large networks and distant initial pages, this could become a major bottleneck in performance.

Future Work

We believe the intersection of graph traversal and recommendation algorithms holds many interesting areas of future exploration.

In applied settings, we imagine users' confirmed preferences–whether in a Wikipedia, Netflix, or food choice setting–would provide the the recommendation algorithm with a source of dynamic information. We envision an applied recommendation system where we first apply our existing algorithm–given an initial set of articles, a set of recommended articles is returned. A human user would then cull the unrelated articles from the returned set. Example scenarios would be choosing which articles to read or deciding which movies to watch. We would combine our initial articles and this curated set of recommended articles, creating a new, larger set of articles with which we reapply our recommendation algorithm. Not only would such an approach iteratively improve the results of our existing algorithm, but we could also add a learning component that would take into account the culling behavior of our users.

A weakness of our algorithms is that if a path between two initial articles does not exist, no recommendations will be returned. While this weakness may be generalized to most graph-based recommendation algorithms, in a practical setting we can imagine multi-partite graphs that would decrease the likelihood of a path not existing between two nodes. Such structures would likely require algorithms that are more complex–both in their formulation as well as their runtime.

References

- [1] Avrachenkov, Konstantin, Nelly Litvak, Danil Nemirovsky, and Natalia Osipova. "Monte Carlo methods in PageRank computation: When one iteration is sufficient." SIAM Journal on Numerical Analysis 45, no. 2 (2007): 890-904.

- [2] Bisnik, Nabhendra, and Alhussein Abouzeid. "Modeling and analysis of random walk search algorithms in p2p networks." In Hot topics in peer-to-peer systems, 2005. HOT-P2P 2005. Second International Workshop on, pp. 95-103. IEEE, 2005.

- [3] Haveliwala, Taher H. "Topic-sensitive pagerank." In Proceedings of the 11th international conference on World Wide Web, pp. 517-526. ACM, 2002.

- [4] Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. Vol. 1. Cambridge: Cambridge university press, 2008.

- [5] Richardson, Matthew, and Pedro Domingos. "The Intelligent surfer: Probabilistic Combination of Link and Content Information in PageRank." In NIPS, pp. 1441-1448. 2001.

- [6] Robert West and Jure Leskovec: Human Wayfinding in Information Networks. 21st International World Wide Web Conference (WWW), 2012.

- [7] Robert West, Joelle Pineau, and Doina Precup: Wikispeedia: An Online Game for Inferring Semantic Distances between Concepts. 21st International Joint Conference on Artificial Intelligence (IJCAI), 2009.

Collaboration Statement:

Both teammates worked collaboratively on all parts and put in equal amounts of effort.