COMMUNITY-PAGERANK: AN EFFICIENT PIPELINE TO PREDICT COMMUNITIES TO USERS

Based on *Beyond Friendship and Followers: The Wikipedia Social Network*

by

Johanna Geiss et al

Vaibhav Sinha, Vishwak S., Chandra Kiran Evuru
CS15BTECH110{34, 43, 12}

April 17, 2018

Review of earlier work

- **Goal:** Extract a large-scale person-centric network structure form English Wikipedia.
- Persons mentioned on a Wikipedia page have a common context, closer the names more evident the relationship.
- Wikidata contains data about 1.2M persons.
- Extracting persons based on Inter-Wiki Link (IWL) and drawing edges based on co-occurrences and distance, obtain a person network of roughly 800k persons and 67M edges.
- Identify centrality, clustering coefficients and component sizes.
- Also identify interesting communities and evaluate them .

Recognition of persons in Wikipedia

- Wikipedia (WP) contains 5.29M content pages, about 65M sentences. [1]
- Classify a WP page as a person page based on year of birth and death, results in 1.05M pages.
- To find references to persons in all Wikipedia page, a two step process is used, first following IWLs and then searching for recognized person names.
- English WP has 76.8M IWLs of which 13.6% (10.4M) refer to persons. 99.9% of these are identified by WD.
- Search each page for persons that are already referred to in an IWL on that page.
- About a third of pages that link to a person page also contain references to persons outside IWLs.

[1] As on January 12, 2015

Following interwiki links (IWLs)

IWL is of the form [[*linkTarget*|*coveredText*]], where *coveredText* is optional.

Algorithm 1 determine person links

```
for each IWL in Wikipedia do:
    linkTarget ← first part of IWL
    if linkTarget == link in Wikidata person entry then
        IWL links to person
    else if linkTarget is a redirect then
        if redirect == link in Wikidata person entry then
            IWL links to person
        end if
    else
        get categories of linkTarget
        if category matches "^\d+(s)? (BC )?births|deaths$"
    then
            IWL links to person
        end if
    end if
end for
```

Person-Centric Network Construction

- The network is represented in the form of a bipartite graph $G = (V \cup D, E)$, where
  - V : Set of nodes corresponding to people.
  - D : Set of nodes corresponding to the Wikipedia documents.
  - E : Set of edges (u, p) such that $u \in V$ and $p \in D$ and iff document p contains person v.
- To construct a network of persons, we project graph G onto the set of persons.
- We create a multi-graph as projection for each co-occurence between two persons as a single edge and then aggregate the edges to obtain a graph w/o multiple edges.

Person-Centric Network Construction

- Weights of edges in the multi-graph are given by :

$$\varphi(e = (v, w, i)) := e^{-d(v,w,i)/2} \tag{1}$$

  - v, w : Belong to the set of persons.
  - i : Instance of a co-occurence between v and w
  - d : Distance between v and w measured as the number of sentences between v and w in the documment that corresponds to i'th co-occurence.

- To convert into a graph with single edges b/w two persons we aggregate multiple edges by cosine similarity of neighbourhoods for the two nodes.

$$dicos(v, w) = \frac{\sum_{e \in n_v \cap n_w} \varphi(e)^2}{\sqrt{\sum_{e \in n_v} \varphi(e)^2} \sqrt{\sum_{e \in n_w} \varphi(e)^2}} \tag{2}$$

Network Properties

- The aggregated network contains exactly $67,583,553$ edges, which connect the $799,181$ persons of which $\approx 98.8\%$ is contained in one giant component / cluster.
- Such a large network will cause inefficient future-processing such as community detection or finding network centrality.
- Based on 5 structural metrics, choose a threshold of 0.0019 for edge weights to retain the edges in the graph.
- `dicos` weighting helps by minimizing outliers and shortening the long tail.
- The authors finally end up with a graph having $\sim$ 65 M edges and $\sim$ 800K nodes.

Finding network centrality

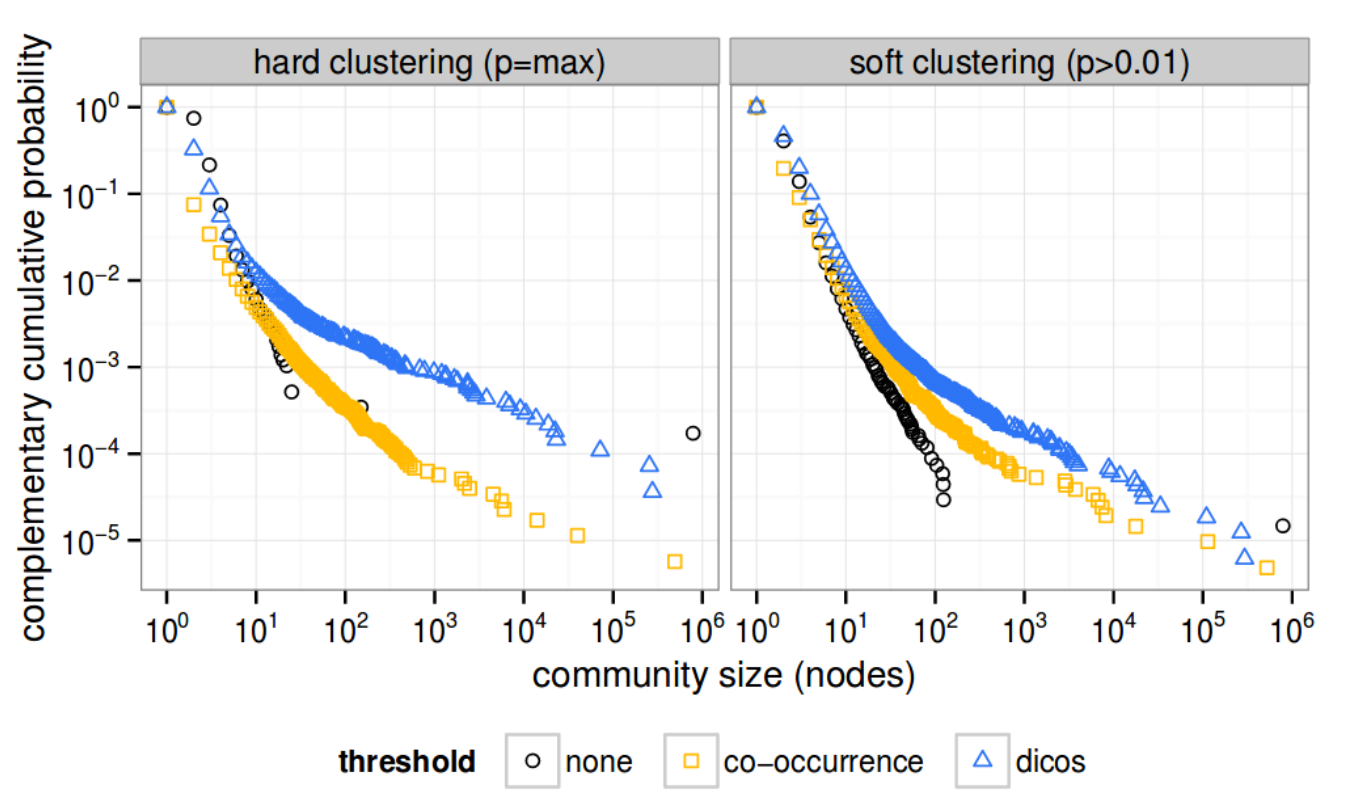
- The PageRank centrality is used to find out the centrality of the network, which in this case is a group of people in the network.

TABLE III. THE 20 HIGHEST RANKED PERSONS IN THE NETWORK WITH A *dicos* THRESHOLD OF $t_\omega = 0.0019$ ACCORDING TO PAGERANK.

rank	degree	gender	birth	death	label
1	1561	m	1961		Barack Obama
2	1449	m	1920	2005	John Paul II
3	1419	m	1946		George W. Bush
4	1508	m	1889	1945	Adolf Hitler
5	1249	m	1946		Bill Clinton
6	1232	m	1882	1945	Franklin D. Roosevelt
7	1572	m	1769	1821	Napoleon
8	1141	m	1927		Benedict XVI
9	1217	f	1926		Elizabeth II
10	1130	m	1911	2004	Ronald Reagan
11	1317	f	1819	1901	Queen Victoria
12	1154	m	1809	1865	Abraham Lincoln
13	1197	m	571	632	Muhammad
14	1242	m	1600	1649	Charles I of England
15	947	m	1890	1969	Dwight D. Eisenhower
16	1321	f	1533	1603	Elizabeth I of England
17	981	m	1913	1994	Richard Nixon
18	1140	m	1732	1799	George Washington
19	996	m	1858	1919	Theodore Roosevelt
20	1056	m	1879	1953	Joseph Stalin

Community Detection

- To detect the communities, the SLPA (stabilized label propagation algorithm) is used.



Here the number of propagations made through the network is equal to 100. Note that the number of medium sized communities discovered is higher for the network obtained using `dicos` thresholding.

Evaluation results and some discussion

TABLE IV. F-SCORE, PRECISION AND RECALL FOR DIFFERENT SETS OF COMMUNITIES

communities	threshold	# comms	# persons	F	P	R
subset	none	90	1,562	**0.4612**	**0.6138**	**0.4341**
10<n<500	t_c	301	10,683	0.4105	0.5583	0.4078
	t_ω	713	24,315	0.3889	0.4785	0.4238
all	none	4,292	798,777	0.2883	**0.6223**	0.2316
	t_c	3,584	677,880	0.2923	0.6002	0.2467
	t_ω	8,193	788,279	**0.2954**	0.5811	**0.2535**

- This is entirely dependent on the policies of categorization which keeps changing.
- This is also dependent on the scheme of thresholding used.
- Hierarchical organization of categories in Wikipedia is a factor to be taken into consideration : to upto what level of hierarchy should we consider?
- Semantically equivalent groups were identified, yet Wikipedia did not have that as a category.

Our contributions

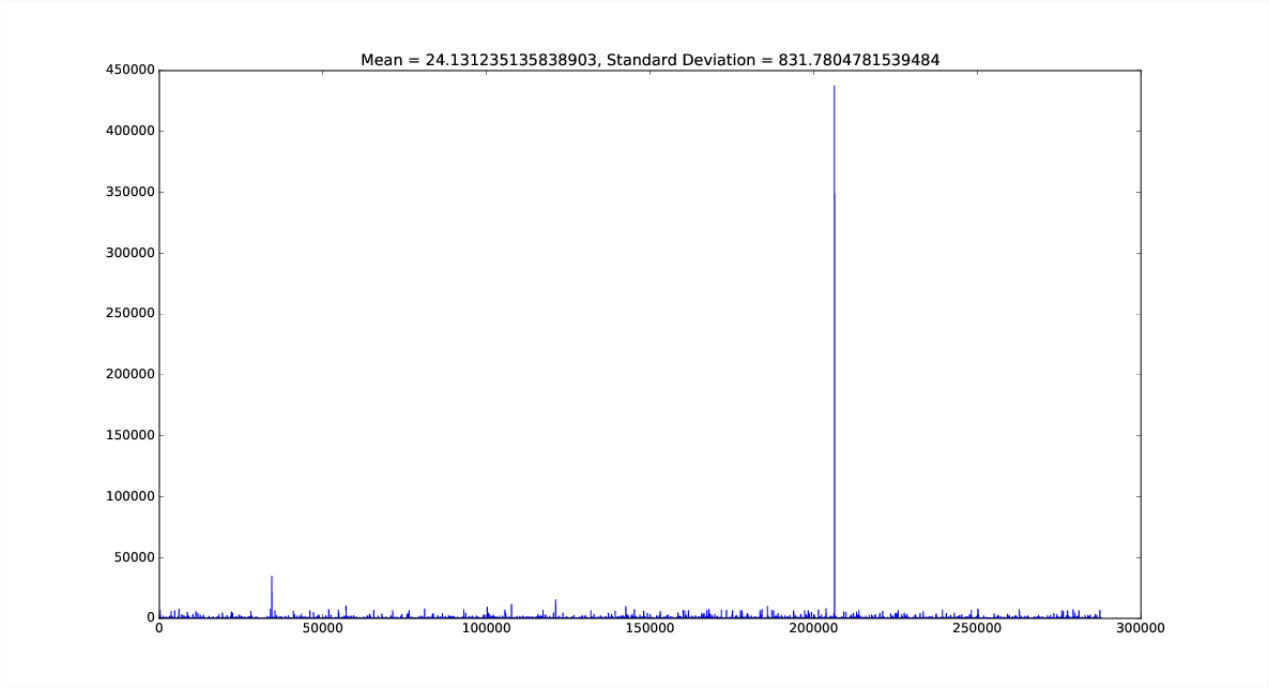
- Verification of statistics presented in the graph - both full and thresholded.
- Verification of the results presented in the paper on the WSN (Wikipedia Social Network) - where we identify the top people from the network, detect communities in the network.
- Community-PageRank Pipeline : Identifying top people in communities, and recommending new pages a community they “might” belong to from the communities that were detected.
  - We test this on three versions of the original graphs - discussion later...

Verification of statistics of the graph

- The paper presents several statistics on the graph like the distribution of genders, occupation, year of birth e.t.c.
- Obtained gender distribution (84.09%, 15.81%) is close to (84.3%, 15.6%).
- Percentage of data having data birth: obtained 82.23%, while reported value 83%.
- Similarly even for distribution of occupation we are able to get similar results.
- Minor deviations from the reported value is probably from the difference in the graph used by the paper and the dataset which we worked with.

Visualizing the dataset

Figure: Variation of the size of categories



Verification of network centrality

- We run PageRank algorithm on the thresholded, weighted graph.
- Among the top 20 reported in paper we get 16 same persons in top 20 in our experiments, the rest 4 are within top 30.
- The exact order among the top-20 is changed, but not drastically.

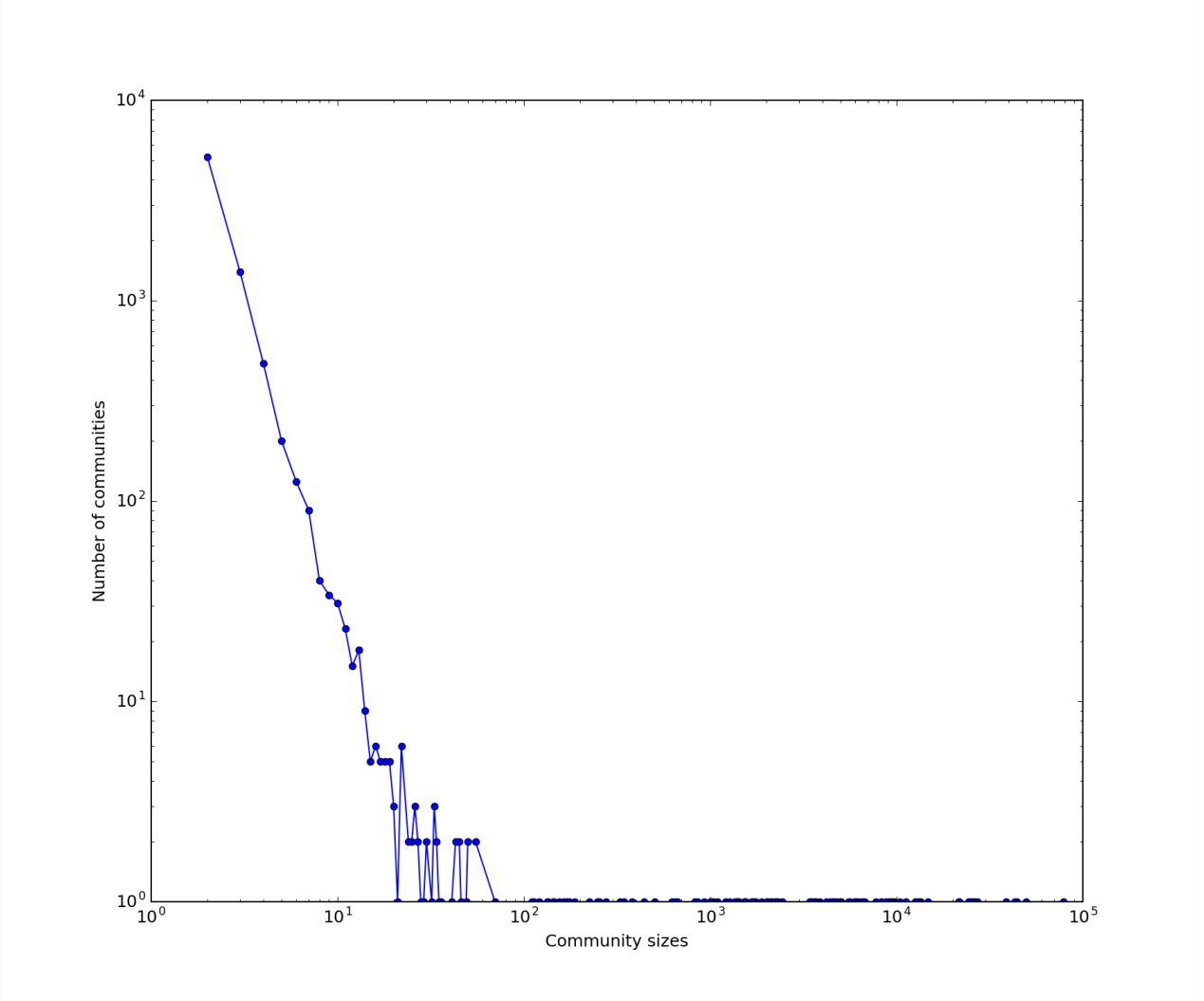
Communities within the graph

- We used Louvain's algorithm for community detection which forms hard clusters, as opposed to SLPA which is a soft clustering algorithm, due to compute constraints.
- We run the Louvain's algorithm on the thresholded, weighted graph.

	All communities	Natural Communities ($10 \leq n \leq 50$)
F1 Score	0.3202	0.3985
Precision	0.8811	0.7661
Recall	0.3547	0.7943
No. of communities	7875	146

Communities within the graph

Figure: Variation of the number of detected communities against size



Community-PageRank Pipeline : Motivation

- Given a page, one would like to assign categories to it.
- There are two methods:
  - Extreme-classification: Assign multiple classes to a page from a wide range of classes.
  - Community-common-categories: Assign a community to a page, get the common categories that the people belong to in that community, and assign it.
- Advantages are that the number of actual classes (here, communities) are relatively low, and is capable of doing multi-category recommendation without as much hassle as extreme classification.
- Novel too!!

Community-PageRank Pipeline

- Detect communities in the graphs and obtain reasonably sized communities i.e., communities that are not too small.
- Obtain the top people in these communities and their pages.
- Use language processing techniques to extract features and subsequently, build a classifier.
- Given a new page, predict a community.

Community-PageRank Pipeline - Community Detection and Filtering

- We require a scalable algorithm considering the size of the graph.
- Due to this, we stuck with Louvain's algorithm as used for the verification of results.
- In our case, we decided from the distribution of the community sizes that minimum size of a reasonably sized community will be 50.
- Hence we filter out the community whose size is less than 50.

Community-PageRank Pipeline - Top People in the Communities

- We get the top 45 people per community in the filtered out communities.
- Why 45?
  - Having too low a value will not accommodate for diversity.
  - Having too high a value might corrupt the quality of the features in later phases.
- We use PageRank for this purpose.

Community-PageRank Pipeline - Natural Language Processing

- Before classification, we need to clean up the documents and extract features from it.
- We do the following preprocessing:
  - Tokenizing the documents.
  - Conversion to lower case.
  - Removal of punctuation.
  - Removal of stop words.
  - Lemmatizing the tokens.
  - Removal of empty tokens.
- To obtain the features from the documents, we now use TF-IDF.

Community-PageRank Pipeline - Prediction for a page

- Now that we know how a page's features are being extracted, we obtain the subset of the entire bunch of features to train a classifier.
- Considering the sparsity and the non-integral values of the features, we didn't resort to using any tree-based classifier (Decision Trees, Random Forests).
- We had 5 candidates:
  - Multinomial Naive Bayes Classifier
  - Multi-layer Perceptions (Shallow Neural Networks)
  - Multinomial Logistic Regression
  - One-vs-All classification with the base classifier as a Binary Logistic Regression model
  - One-vs-One classification with the base classifier as a Binary Logistic Regression model

Our Results

- We assemble our training set as the pages of the top 45 users in each community.
- The test set is obtained by taking a random 5 among the rest of the users in every community.
- We do this for three variation of the graphs and community subgraphs

S1 Community Detection and PageRank with the weighted graph
S2 Community Detection with the weighted graph and PageRank on the unweighted graph
S3 Community Detection and PageRank with the unweighted graph

Scheme	Number of "valid" communities	Train Set	Test Set	
S1	106	4680	530	$\Rightarrow$ D1
S2	106	4511	530	$\Rightarrow$ D2
S3	54	2317	269	$\Rightarrow$ D3

Our Results ... contd..

Our best and worst accuracies achieved on these datasets are tabulated below:

Dataset	Best Test Set Accuracy	Worst Test Set Accuracy
D1 (with 20000 features)	66.038%	47.925%
D2 (with 20000 features)	68.491%	45.472%
D3 (with 10000 features)	63.941%	47.955%

- The best results were achieved using Multinomial Logistic Regression, One-vs-All Binary Logistic Regression.
- The worst results were achieved using K-Nearest Neighbours and a 2-hidden layer multi-layer perceptron (optimized using L-BFGS).

Our Results ... contd..

- To test the efficacy of TF-IDF and also to establish certain baselines, we obtain three more set of features from each dataset.
- These three variants are obtained using top-K term frequencies (TF) and K-randomly picked features from the computed TF-IDF and TF as opposed to best features.
- The top-K TF-IDF are obtained by checking those K features, which are more correlated with the result.
- We proceed to train the classifiers with these sets, and the best obtained results are tabulated below.

Dataset	K	Best - TF-IDF	Best - TF	Random - TF-IDF	Random - TF
D1	20000	66.038%	64.717%	10%	8.113%
D2	20000	68.491%	62.830%	8.868%	7.547%
D3	10000	63.941%	60.595%	10.781%	10.409%

Acknowledgements

We would like to thank Nagendra Kumar for his valuable guidance and motivation throughout the course of the project.

Questions?

Thank You.

Our results are reproducible, so please check out
https://github.com/vishwakftw/CS6670-TDM
for the code