How Does Popularity Diffuse in a Wikipedia Sub-Graph?

The Wikigraph

Starting from an initial Wikipedia Page, this group of functions built the weighted graph of depth 2 (we consider only links with multiplicity > 1). Multiplicity is our weight.

Built the Graph

```
initialTitle = "Star_Trek_Into_Darkness";
```

```
Needs["GraphUtilities`"]
```

```
cleanlinks[links_] := Module[
    {cleanedlinks},
    cleanedlinks = Select[
        links,
        StringMatchQ[#, "http://en.wikipedia.org/wiki/"~~Except[":"]..] &
    ];
    cleanedlinks = Select[ Gather[DeleteCases[cleanedlinks,"http://en.wikipedia.org/wi
    Tally@Flatten[cleanedlinks]
    (*First /@ cleanedlinks*)
]
(*this function will consider links multiplicity and links with at least 2 repetitions
```

```
importFirstLevelLinks[init_]:= cleanlinks @
    Import["http://en.wikipedia.org/wiki/"<>init, "Hyperlinks"]
(* this output is a list of weighted links with their multiplicity *)
```

```
createFirstLevelWeightedEdges[init_,weightedlinks_]:=
{init,Last@StringSplit[#[[1]],"/"],#[[2]]}& /@ weightedlinks
```

```
findFirstNeighbors[firstLevelLinks_]:= Last@StringSplit[#,"/"]& /@firstLevelLinks[[All
```

```
importSecondLevelLinks[firstLevelLinks_]:= (cleanlinks @ Import[#,"Hyperlinks"])& /@ f
```

```
createSecondLevelWeightedEdges[firstNeighbors_,secondLevelLinks_]:=
    Module[{secondNeighbors,secondNeighborsWeighted},
        secondNeighbors=Map[Last[StringSplit[#,"/"]]&,secondLevelLinks[[All,All,1]],{2
        secondNeighborsWeighted=Transpose[{secondNeighbors[[#]],secondLevelLinks[[#,Al
        Flatten[Outer[
            Flatten[{#1,#2}]&,{firstNeighbors[[#]]},{secondNeighborsWeighted[[#]]},2]&
]
```

```
createTheGraph[init_]:=Module[{firstLevel,secondLevel},
firstLevel=importFirstLevelLinks[init];
secondLevel=importSecondLevelLinks[firstLevel];
Flatten[{createFirstLevelWeightedEdges[init,firstLevel],createSecondLevelWeightedEdges


]
```

Remove Self Loops

```
NoSelfLoopGraph[EdgesList_]:= DeleteCases[EdgesList,{a_,a_,__}]
```

Correct Multiplicity: remove Multi Edges and update Weights

```
NewGraphWithCorrectMultiplicity[Edges_]:=Module[{unweighted,repetitions,positions,
replaceIndices,deleteIndices,updates,subs,checkedFirst,checkedFinal},
unweighted=Edges[[All,;;2]];
repetitions=Select[Tally[unweighted],(#[[2]]>1)&];
positions=Position[unweighted,#[[;;2]]]&@@@ repetitions;
replaceIndices=positions[[All,1]];
deleteIndices=positions[[All,2]];
updates=Partition[Flatten[{Edges[[#1[[1]],;;2]],Edges[[#1[[1]],3]]+Edges[[#1[[1]],3]]}
subs=Map[#1[[1,1]]→#[[2]]&  ,Transpose[{replaceIndices,updates}]];
checkedFirst=ReplacePart[Edges,subs];
checkedFinal=Delete[checkedFirst,deleteIndices]

(*this function controls and corrects repetitions in the graph, just use it once*)
]
```

Operations on the Graph

Static visualization with Weights or Ranks

```
FancyRankingGraph[list_,ranks_,title_,center_]:=
    Graph[DirectedEdge@@@ list,
        VertexSize→ranks,GraphHighlight→{center},
        PlotLabel→Framed[Style[title,15,"Arial"],
        Background→LightBlue],VertexSize→1,ImageSize→700,GraphLayout→"SpringElectrica
        VertexLabels→{center→Framed[Style[center,15,"Arial"],Background→White]}
]
```

```
NormRankings[rankings_]:=
With[{mean=Mean[rankings[[All,2]]]},

(#→N[#2/mean]& @@@ rankings)
]
```

Dynamic visualization with Popularity

```
AssignDynamicWeights[Vertices_,TimeSeries_]:=
    MapThread[#1 → Flatten[#2[[All,All,2]]] &, {Vertices,TimeSeries}]
```

```
PopWeightedGraph[list_,weights_,title_,center_]:=Graph[DirectedEdge@@@ list,
VertexSize→weights,GraphHighlight→{center},
PlotLabel→Framed[Style[title,15,"Arial"],
Background→LightBlue],VertexSize→1,ImageSize→700,GraphLayout→"SpringElectricalEmbeddi
VertexLabels→{center→Framed[Style[center,15,"Arial"],Background→White]}]
```

```
DynamicPopularity[Edges_,popweights_]:=Manipulate[
 With[
 {mean = Mean[Flatten[popweights[[All, 2]]]]},
 PopWeightedGraph[Edges,#1→N[#2[[w]]/ mean]&@@@ popweights
 ,"Avatar, popularity","Avatar_%282009_film%29"]
 ]
 ,{{w,1},1, Length[popweights[[1,2]]], 1}]
```

Popularity Time Series

Extract Time Series for a set of Wikipedia Pages

Where? http : // stats.grok.se/json/en/

Given a list of Titles of Wikipedia Pages this function extract the time series for the chosen dates (in months), and print the results on a json file.

```
importingOnFile[dates_List,pages_,outputfile_]:=
Do[
    (Import["http://stats.grok.se/json/en/"<>#1<>"/"<>pages[[i]], "JSON"]>>>
        "/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/"<>outputfile) & /@ dat
    ,
    {i,Length[pages]}]

(*dates_List = {"200910","200911","200910",..} in months, in crescent order, pages = l
```

Process and Plot Time Series

This function read from file the time series, knowing how many months.

```
readingTimeSeries[file_,months_]:=ReadList["/Users/Levantina/Documents/WOLFRAM/PROJECT
```

```
CleanTimeSeries[series_]:={FromDigits /@ StringSplit[#,"-"],#2}& @@@ series[[1,2]]
```

This function cleans imported data and makes them ready to be plotted.

```
ExtractTimeSeries[imported_List,Nmonths_Integer]:=Partition[CleanTimeSeries[#]& /@ imp
```

This function is useful to study the average behaviour of a random sample of pages.

```
averageTimeSeries[cleanedTS_]:=N@Mean[Flatten[cleanedTS[[#,All,All,2]]]& /@ Range[1,Le
```

This function plots the popularity for the selected Vertex:

```
PlotTimeSeries[series_,opts___]:= DateListPlot[series,Joined →True,opts]
```

```
ShowTimeSeries[index_,vertices_,imported_List,Nmonths_Integer]:=
    PlotTimeSeries[
        ExtractTimeSeries[imported,Nmonths][[index]],
            PlotRange→All,PlotLabel→vertices[[index]]
        ]
    ]
```

This function correct the weekly fluctuation effect:

```
Corrections[daily_,timeseries_]:=
Map[#/Flatten[Table[daily,{9}]][[;;Length[#]]] &, timeseries,{1}]
```

```
ex = Import["/Users/Levantina/Documents/WOLFRAM/PROJECT/startrekNetwork/
      allStarTrekWeighted.tsv", "TSV"];
```

```
Length[NewGraphWithCorrectMultiplicity[ex]]
```

```
3126
```

Correlations between Time Series

```
FastCorrelations[timeseries_]:=With[{matrix=If[
    StandardDeviation[#]>0,N[(#-Mean[#])/(StandardDeviation[#]*Sqrt[Length[#]])],
    Table[0.,{Length[#]}]]&/@timeseries},
    matrix.Transpose[matrix]
]
```

We can transform the correlation in a distance in the graph:
d(i,j) = Sqrt[2*(1-corr(i,j))]
d= {0 correlated, Sqrt[2] not correlated, 2 anticorrelated}

```
DistancesFromCorrelationMatrix[matrix_]:= Sqrt[2*(1-#)] & /@ matrix
```

Adjacency Matrix

To put 0 s on the diagonal (for Array Plot)

```
removingSelfLoops=(-1)*(IdentityMatrix[65]-1);
```

To visulaize with the function Graph, to put Infinity to disconnect vertices

```
InfiniteDiagonal=Map[ If[#⩵1,Infinity,1]& ,IdentityMatrix[65],{2}];
```

Link Prediction

```
LinkPrediction[d_,popi_,popj_,DistThreshold_,PopThreshold_]:= If[d>DistThreshold,
d*Sqrt[popi*popj],
If[popi*popj>PopThreshold,d*Sqrt[popi*popj],0]]
```

```
GraphPrediction[adjMatrix_,popularities_,DistThreshold_,PopThreshold_]:=Map[LinkPredic
popularities[[#[[1]]]],popularities[[#[[2]]]],DistThreshold,PopThreshold]&,
Table[{i,j},{i,1,Length[adjMatrix[[1]]]},{j,1,Length[adjMatrix[[1]]]}],{2}];
```