DS8009 Project- Analyzing Company Relationships on the Web

Rafik Matta

December 9, 2018

Contents

Introduction and Problem Description

In today's business world, a company's Web presence is more critical than ever. It's so critical that an entire industry now exists to allow companies to optimize for customer's discovering them through the many portals that customers use to browse the internet including search engine optimization for search engines like Google or Bing or through advertising campaigns on social networks like Facebook or Twitter. Many of the largest companies have been for the at least the last two decades working on improving their web presence as they've determined how critical it is to their overall success [1]. Given that the modern Web has been around for over 20 years now, many companies have perfected their ability to bring customers to their sites, inform them about their business, sell to those customers, and guide them through their entire business life cycle [2].

There is now enough data on these different sites, that we can begin to further explore companies and their relationships. Although the effects of networks have been previously studied, attempting to correlate real-world performance of an organization with their web presence is a relatively unexplored form of analysis. If we are given enough information about how companies interact with each other on the web and combine it with our knowledge of their real-world relationships and rankings, can we potentially predict future relationships that may form? There aren't currently any open data sets to explore this particular problem as we would require a lot of historical information which hasn't been gathered as of yet. What is available and relatively unexplored is a recent enough snapshot that insights should be gatherable to do some meaningful analysis. This was done in a prior study that is also referenced later on [3]. Using a different dataset and different methodology, we will attempt to replicate a similar form of analysis to that study and answer some extension questions. Some interesting questions that we might be able to answer:

1) Are Internet-first companies (those who's primary business model is internet based) the most influential companies on the Web?
2) Is there any relationship between those companies' industries/size to their influence on the Web?
3) Can we infer those relationships from how these companies refer to and link to each other on their websites?
4) What relationships do these influential companies have with other large influential companies?
5) What do these relationships mean? Can they potentially be used to predict further relationships?

Methodology

To perform this analysis as accurately as possible and to allow for scrutiny, some baseline assumptions need to be made clear.

Assumptions and Restrictions

1) An Internet-first company is any company whose primary business model is about interacting with their customers via the Web or supplying to companies' whose primary business model is Internet based.
2) As the data is rather large, the analysis will focus on the companies that meet the above criterion as well as being considered influential and large in the real-world.
3) Generally speaking, a company whose primary business model includes software or software sales will be primarily dependent on the Internet.
4) Software companies are classified as any company that packages and distributes software as a vendor through any medium (for example Google is a search engine + software company due to Google Cloud, whereas Facebook is not)

With the above in mind, we focus the analysis strictly on the companies that fit the above criteria and see how they fare with their Web presence and influence on the Web. This is done to the exclusions of analyzing the influence of other companies that are not seen as large real-world companies and those companies are almost strictly used in broader analysis of the graph data or deeper analysis on the largest companies.

Techniques

Pre-processing had to be done on the dataset that is used (and further explained in the Dataset section) in order for it to be useful. Much of this pre-processing was culling incomplete data and reformatting the data to make it appropriate for consumption by the software used for analysis. No further pre-processing was done to the data.

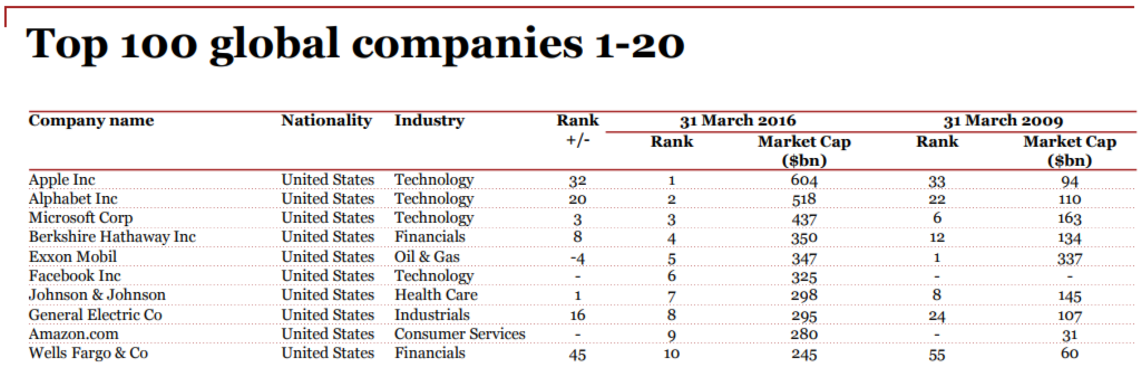
As the data available for the analysis is primarily website link data, the analysis makes heavy use of the following:

1. Community Detection [4] [5]
   As with most graph analysis, there tends to be natural communities that form. This is especially true on the Web. Companies will have multiple domains and sub-domains that link to each other, and almost certainly most companies that have relationships with each other will link to each other somehow. For community detection both Connected Components and Label Propagation were used. Connected components was used as a pre-processing step to see how connected the graph is. Label Propagation was used as a point of comparison.
2. Node Centrality [6]
   As the analysis aims to focus on finding a correlation between real-world size/influence and Web size/influence, node centrality is at the heart of the analysis. PageRank was used to determine who the most influential companies were, based on the types of relationships their domains had with other companies' domains.
3. Node Similarity [7] [8]
   In order to further analyze how these influential companies, interact, both the Jaccard Similarity algorithm and Overlap similarity algorithm were contrasted and used to determine node similarity amongst the largest companies' domains to see how similar they are and if any further study is warranted to determine if predictions can be made amongst them.

Determining Large Companies

It’s important to establish some base level knowledge about the corporations that will be the subjects of this analysis. As will be seen in later sections, many of the companies that are prominently featured tend to be rather large. As the Web is a global network with equal opportunities of access in most developed Internet markets, it is fair to assume that this can be treated as a global analysis. As such we will focus on gathering prior knowledge about global companies.

Figure 1 shows the 10 largest companies in the world (size being relative to market capitalization) as of 2016. As expected, not necessarily all of these companies are in the Technology sector but Technology features prominently.



Top 100 global companies 1-20

Company name	Nationality	Industry	Rank +/-	31 March 2016		31 March 2009	
				Rank	Market Cap ($bn)	Rank	Market Cap ($bn)
Apple Inc	United States	Technology	32	1	604	33	94
Alphabet Inc	United States	Technology	20	2	518	22	110
Microsoft Corp	United States	Technology	3	3	437	6	163
Berkshire Hathaway Inc	United States	Financials	8	4	350	12	134
Exxon Mobil	United States	Oil & Gas	-4	5	347	1	337
Facebook Inc	United States	Technology	-	6	325	-	-
Johnson & Johnson	United States	Health Care	1	7	298	8	145
General Electric Co	United States	Industrials	16	8	295	24	107
Amazon.com	United States	Consumer Services	-	9	280	-	31
Wells Fargo & Co	United States	Financials	45	10	245	55	60

Figure 1. Top 10 Companies by Market Cap. *[9]*

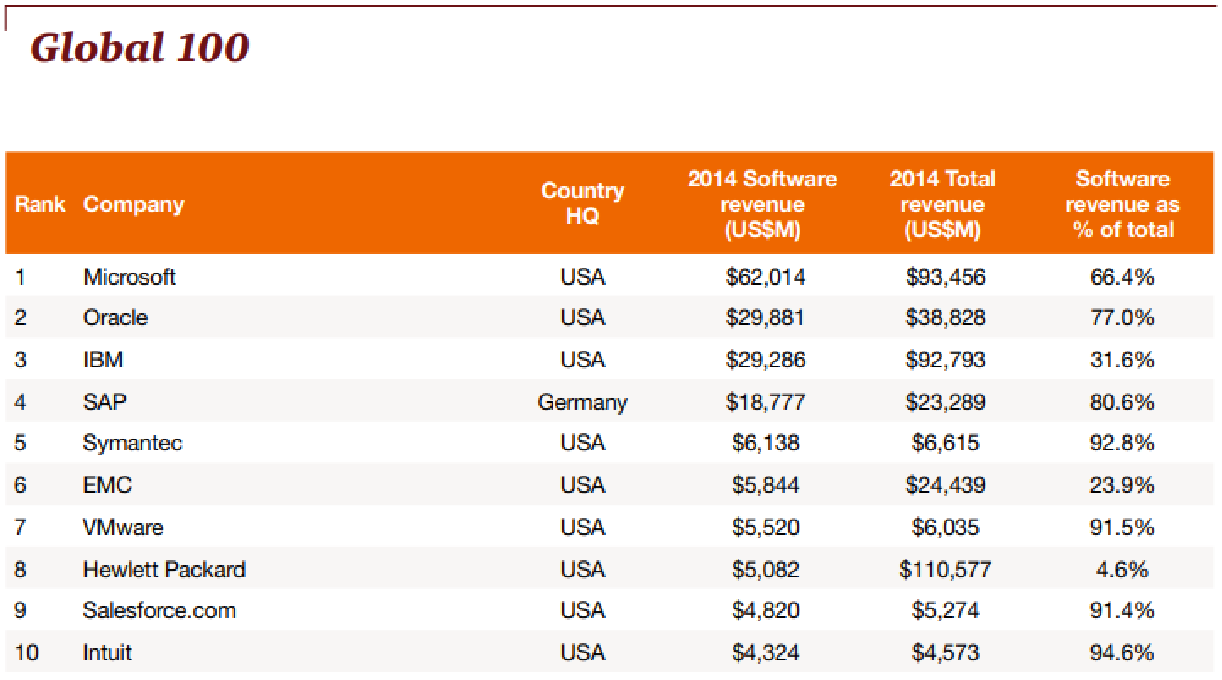
It is further important to know who the largest Technology or Internet-based companies are. As of 2016 (which is the year our dataset is most relevant for) the top technology companies are:

Company name	Nationality	Industry	Rank	Market Cap($bn)
Apple Inc	United States	Technology	1	604
Alphabet Inc	United States	Technology	2	518
Microsoft Corp	United States	Technology	3	437
Facebook Inc	United States	Technology	6	325
Amazon.com	United States	Consumer Services	9	280
Alibaba Group Holding	China	Consumer Services	23	196
Tencent Holdings	China	Technology	26	192
Oracle Corp	United States	Technology	31	170
Home Depot	United States	Consumer Services	32	167
Walt Disney Co	United States	Consumer Services	34	162
Intel Corp	United States	Technology	37	153
IBM Corp	United States	Technology	43	146
Cisco Systems	United States	Technology	44	143

Figure 2. Largest Technology/Internet Companies as of 2016. *[9]*

As mentioned in the Methodology section, it’s generally assumed that companies that are considered primarily Software Companies have a heavy reliance on the Internet as well. Interestingly many of the companies considered Technology companies happen to also be considered Software companies as they might be involved in software distribution or offer a software-as-a-service (SaaS). Therefore, we also look at who the dominant

Software companies are. Figure 2 shows us who the top 10 largest Software companies are as of 2016.



Global 100

Rank	Company	Country HQ	2014 Software revenue (US$M)	2014 Total revenue (US$M)	Software revenue as % of total
1	Microsoft	USA	$62,014	$93,456	66.4%
2	Oracle	USA	$29,881	$38,828	77.0%
3	IBM	USA	$29,286	$92,793	31.6%
4	SAP	Germany	$18,777	$23,289	80.6%
5	Symantec	USA	$6,138	$6,615	92.8%
6	EMC	USA	$5,844	$24,439	23.9%
7	VMware	USA	$5,520	$6,035	91.5%
8	Hewlett Packard	USA	$5,082	$110,577	4.6%
9	Salesforce.com	USA	$4,820	$5,274	91.4%
10	Intuit	USA	$4,324	$4,573	94.6%

Figure 3. Top 10 Software Companies *[10]*

Dataset

Name: Relato Business Graph Database
Link: https://data.world/datasyndrome/relato-business-graph-database
Relevance of Data: April, 2016

As per the description on the above site:

> "It contains links between businesses pulled from the web. It contains 373,663 links between companies, of the type's "partnership" (one company listed on another company's partnership page), "customer" (one company listed on another company's example customer page), "competitor" (co-bidders on AdWords above some limit), "investment" (a company listed on a VC's website), "supplier" (the inverse of the "customer" type. This dataset was used to drive both a lead generation system where metrics on the graph fed into a classification for leads (lead/no lead) and a market visualization system (a force directed layout of markets and their segments)."

Interestingly enough, the main purpose of this graph was to serve as data for a lead generation system. This means that they must have weighted relationships in a specific way (which is not present in the dataset). Applying this type of weighting to the relationships could be used to perform interesting analysis that is not in the scope of the problem definition but would make an interesting follow on study.

After pre-processing, and removal of some incomplete link data, here are some basic network statistics:

$$|V| = 51{,}222$$

$$|E| = 373{,}542$$

The dataset is split into two JSON files. Figures 4 and 5 show and describe the attributes as well as explain their usage or lack thereof in the analysis.

companies.json

Attribute	**Description**	**How Used**
id	an assigned id	Not Used
update_time	the last update made to the data	Used to determine data relevance
username	Name of the user who update the data	Not used
name	Company name	Used to Associate domains with Companies
domain	Company domain	Used for Node Labels

Figure 4 Companies Data Model

links.json

Attribute	**Description**	**How Used**
id	an assigned id	Not Used
update_time	the last update made to the data	Used to determine data relevance
home_name	Name of the company's website that was being analyzed	Not used
home_domain	Domain of the company's website that was being analyzed	Used for Node Labels
link_name	Name of the company's link that was on the website being analyzed	Not used
link_domain	Domain of the company's link that was on the website being analyzed	Used for Node Labels

Figure 5 Links Data Model

Technologies Used & Pre-processing Steps

Technologies

Technology choice for a given problem is always an interesting challenge. Many factors have to be considered as many tools can accomplish the same thing but have different requirements.

The analysis was performed on the following system:

- Windows 10 - 64-bit
- 16GB of DDR3 RAM
- Intel Core i7 Quad-Core with Hyperthreading (early 2013 model).
- Samsung EVO SSD

The following is the list of technologies used along with their purposes:

1) Json2csv NodeJS package – use to convert JSON data into csv data
2) Excel – used extensively for data pre-processing as well as data and results validation
3) Neo4j – used for most of the actual analysis and visualizations produced

Initially, the intention was to actually use Spark and GraphX. The major obstacle here was the learning curve associated with Scala, as GraphX does not yet support a Python API. Another option was to use Python + the NetworkX library. No complete empirical analysis has been done as it is not within the scope of this assignment, but preliminary results shows that Python+NetworkX performed exceptionally poorer as compared to Neo4j. This is likely attributable to Python's serial nature as an interpreted dynamically typed language. Further, producing graph visualizations, especially for large networks is a major shortcoming of current libraries available for Python.

As can be seen from the description of the dataset in the previous section, the graph isn't too large but is still rather sizeable. As such, it became apparent that Neo4j offered the best compromise. Neo4j is relatively easy to use, with a query interface reminiscent of SQL. It offers optimized implementations of most graph algorithms that are multithreaded allowing for quick performance. Neo4j does require a large amount of available memory, so a larger dataset would have potentially caused issues.

Preprocessing

The dataset described in the previous section required some basic pre-processing to make it useful.

1) Convert links.json to csv using json2csv
2) Remove rows without a labeled link "type"
3) Remove rows that have any of the chosen analysis attributes being NULL
4) Load data into neo4j

Analysis and Results

The graph contains 51,222 nodes. A node in the graph is a web domain which belongs to a company. There are 373,542 relationships. As this is a directional graph, it's clear that it can actually be seen as quite sparse. The total possible edges are 2,623,642,062. The total graph density is shown in Equation 1.

$$D = \frac{|E|}{|V|(|V|-1)} = 1.424\mathrm{E}-4$$

Equation 1. Graph Density Calculation

As the graph is not a particularly dense one, to further analyze this graph and see if our questions can be answered we must begin by understanding the different types of relationships between nodes in our graph. This can be seen in Table 1.

Link Type	**Link Description**	**Count**
Partnership	One company listed on another company's partnership page	112,948
Competitor	Co-bidders on AdWords above some limit	29,100
investment	A company listed on a VC's website	71,628
Customer	One company listed on another company's example customer page	80,465
Supplier	Inverse of customer	79,401

Table 1. Relationship Types

As previously mentioned, the graph is directional. Each link goes from a home domain to a link domain. The graph is unweighted. That means each link has equal weight and the number of in/out connections is what matters in further analysis. That being said, the graph could be weighted. Any two company domains can share any of the relationship types and therefore can have multiple edges between them. Depending on the type of analysis we want to do, some types of relationships might be more important than others. As such, a good place to begin to understand how the graph is structured is to perform community detection.

Connected Components vs. Label Propagation

Both Connected Components and Label Propagation were performed on the graph using the implementations available within Neo4j [4] [5]. The results of the analysis can be seen in Table 2. Generally, a competitor relationship is a two-way mutually acknowledged relationship and this can clearly be seen from the Label Propagation result. Alternatively, that mutuality wouldn't necessarily exist in the other types of relationships except for the Partnership type. Interestingly enough, the lack of mutuality shown in the Partnership relationship is indicative of the fact that it can tend to be a bit of a One-to-Many relationship in that, if a larger company has multiple relationships with smaller companies, that relationship may be important enough for those smaller companies to acknowledge, but not necessarily for the larger company to acknowledge.

Relationship Type	Connected Components	Label Propagation (Out/In)
Competitor	41,399	44,765/ 45,300
Investment	39,046	28,958/1,689
Customer	32,836	28,969/1,709
Supplier	33,509	28,959/1,715
Partnership	9,940	28,949/1,676
All	146	28,944/1,696

Table 2. Connected Components vs. Label Propagation

PageRank

Given what Community Detection has shown us, further analysis needs to be done to determine node centrality. As these are web links, the most obvious analysis to determine Node Centrality would be running PageRank on the graph. The Neo4j PageRank Implementation was used [6]. There are different relationship types so this needed to be taken into account. Table 3 and Table 4 shows the results of that analysis and it's clear that there are some dominant nodes or domains in the graph, namely:

- "microsoft.com"
- "google.com"
- "amazon.com"
- "facebook.com"
- "apple.com"
- "ibm.com"
- "oracle.com"

This validates the hypothesis that the largest technology companies are dominant as well on the Web as these domains belong to some of the largest companies in the world, as well as largest software companies as shown in Figure 1 and Figure 3.

page	**score**
"microsoft.com"	188.9347365
"google.com"	179.2081525
"ibm.com"	129.3353715
"facebook.com"	127.9988145
"apple.com"	92.4722315
"hp.com"	88.6145405
"oracle.com"	87.018504
"amazon.com"	79.3858695
"cisco.com"	76.52539
"sap.com"	74.5817715

Table 3. Top 10 domains for entire graph

Table 3 is showing the dominant nodes in the overall graph, but we cannot simply look at this and be satisfied as the relationship type matters as well. Some companies may have domains that are more dominant with certain relationship types than others and this can lead to some interesting conclusions. Table 4 shows PageRank applied to the different relationship types. There are some companies that aren't amongst the top 10 dominant domains of the entire graph. Nevertheless, regardless of relationship type the above 10 companies still feature prominently.

Supplier		Customer		Partnership		Competitor		Investment	
page	score	page	score	page	score	page	score	page	score
"microsoft "	164.85	"google "	77.57	"microsoft "	60.19	"google "	41.42	"google "	39.26
"google "	145.74	"microsoft "	59.90	"google "	39.56	"amazon "	31.05	"microsoft "	32.68
"ibm "	119.61	"facebook "	52.71	"ibm "	36.06	"apple "	26.88	"amazon "	31.34
"facebook "	112.72	"ibm "	41.99	"apple "	24.42	"facebook "	21.54	"apple "	29.72
"oracle "	88.82	"apple "	34.25	"facebook "	21.04	"microsoft "	20.74	"facebook "	27.45
"amazon "	88.60	"hp "	31.51	"cisco "	19.72	"walmartstores "	16.43	"bloomberg "	21.93
"gainsight "	77.53	"sap "	30.41	"hp "	18.97	"sony "	14.26	"ebay "	21.28
"adobe "	76.06	"oracle "	29.29	"oracle "	16.02	"ebay "	13.17	"ibm "	20.53
"hp "	72.45	"cisco "	26.19	"samsung "	15.86	"samsung "	12.87	"ge "	17.11
"sap "	70.23	"intel"	21.57	"intel "	15.86	"ibm "	10.72	"goldmansachs "	16.54

Table 4. Page Rank Results

Jaccard vs. Overlap Similarity Measure for Top 5 Most influential Companies

As the dataset is rather large, in order to make any meaningful measure across all of the Companies both the Jaccard and Overlap similarity measures were used to measure similarity across the top 7 company domains predicted by PageRank to be central to the network. The Jaccard Similarity Algorithm implementation from Neo4J is used [7]. The Overlap Similarity Algorithm implementation is also from Neo4J [8]. Companies that generally fall in the same industry/category have a very strong similarity score. An interesting company in our analysis is Google which scores highly against Microsoft and Facebook. With Jaccard, HP also showed as being rather similar to some of the top companies. This makes sense as in Table 3 and 4 we can see that it is a highly influential node in our graph. As can be seen in Tables 5 and 6, Overlap vs. Jaccard similarity scores are not exactly the same across the entire graph for all the companies, and Overlap similarity clearly shows Microsoft being much more similar to other nodes vs. Jaccard. In this case, given the characteristics of the graph, Jaccard performs better at predicting actual relationships whereas Overlap is better at characterizing the entire graph amongst all types of relationships.

home_domain	link_domain	intersection	similarity
"google.com"	"facebook.com"	2826	0.273757629
"facebook.com"	"google.com"	2826	0.273757629
"ibm.com"	"hp.com"	2217	0.26830449
"microsoft.com"	"google.com"	3537	0.266963544
"apple.com"	"hp.com"	1673	0.253523261
"oracle.com"	"ibm.com"	2270	0.250690226
"amazon.com"	"apple.com"	1474	0.245789561

Table 5. Jaccard Similarity of top 7 Companies against the entire graph.

home_domain	link_domain	intersection	similarity
"apple.com"	"microsoft.com"	2388	0.594029851
"amazon.com"	"google.com"	1988	0.576064909
"ibm.com"	"microsoft.com"	3212	0.515735389
"facebook.com"	"google.com"	2826	0.509280952
"oracle.com"	"microsoft.com"	2427	0.476162448
"google.com"	"microsoft.com"	3537	0.465394737
"microsoft.com"	"cisco.com"	2594	0.28238624

Table 6. Overlap Similarity of Top 7 Companies against entire graph.

When the top 7 companies are analyzed amongst themselves for Jaccard similarity, Google again features prominently as the company with the highest similarity score against most of its peer large companies.

home_domain	link_domain	intersection	similarity
"google.com"	"facebook.com"	2826	0.273757629
"facebook.com"	"google.com"	2826	0.273757629
"microsoft.com"	"google.com"	3537	0.266963544
"ibm.com"	"google.com"	2898	0.265141812
"oracle.com"	"ibm.com"	2270	0.250690226
"apple.com"	"amazon.com"	1474	0.245789561
"amazon.com"	"apple.com"	1474	0.245789561

Table 7. Jaccard Similarity measures for top 7 companies against each other.

home_domain	link_domain	intersection	similarity
"apple.com"	"microsoft.com"	2388	0.594029851
"amazon.com"	"google.com"	1988	0.576064909
"ibm.com"	"microsoft.com"	3212	0.515735389
"facebook.com"	"google.com"	2826	0.509280952
"oracle.com"	"microsoft.com"	2427	0.476162448
"google.com"	"microsoft.com"	3537	0.465394737

Table 8. Overlap Similarity of Top 7 companies against each other.

When looking at Overlap Similarity for just the Top 7 companies, “microsoft.com” again features prominently. This again shows the different characteristics that the two similarity measures show. Although “microsoft.com” is rather prominent in general, when it comes to learning about the relationships between the companies, Jaccard is more reflective of the reality. We can actually rightly use the Jaccard similarity score to determine which companies are strong competitors and whenever a new link is formed with one of these companies, there’s a strong chance that that same link can also be formed with one of it’s competing companies. Not surprisingly, most of these companies share a “competitor” link amongst each other.

Conclusions and Lessons Learned

As can be seen, companies that are generally considered large in the real world are also rather influential on the Web. This further re-affirms the 2015 study done under a similar premise but using a different analysis methodology by Yun & Gloor [3]. It is hard to determine if that Web-based dominance is the result of their real-world dominance or vice-versa. There is certainly a strong correlation between the two, but that needs to be further studied. A further interesting conclusion is the absence of two large technology companies from the top rankings. Both Alibaba and Tencent, who are dominant in the real world as technology companies, do not rank in the Top 20 companies on the Web. This leads to the possible conclusion that the original assumption of the "globalness" of the Web is incorrect and there might actually be competing Web's with little interconnectivity between them.

Using Overlap similarity scoring, we can do a general analysis of domains within our dataset and see who is most similar across all relationship types. This does not unfortunately inform any predictive analysis. On the other hand, using Jaccard similarity scoring we can see how strongly related two companies are and we can correctly infer that the more similar two dominant companies are on the internet, the more likely they are to also be competitors vying for the same business. In other words, a strong Jaccard similarity score between two companies is generally a good indication of how balanced their competitive relationship is. Further it seems that the Web is heavily dominated by a few large players and their web presence extends beyond their own domains. Using Jaccard similarity we can conclude that, if two companies are dominant on the internet and very similar, any new links that form between one of those companies and other non-dominant companies can serve as predictors of future links that will form between the second company and the same non-dominant companies as they tend to be competitors. Further interesting analysis can and should be done using different similarity measures to compare against Jaccard and Overlap for different prediction tasks.

As for the analysis itself, some lessons learned include that not all technology stacks are created equal. There are many tools available to perform network and graph analysis, and they all have their strengths and weaknesses. Neo4j was primarily used. Spark with GraphX could have also been used but the leap from the SQL to the Cypher Query Language requires an overall smaller learning curve than from Python/Java to Scala.

References

[1] N. &. H. Q. Heinze, "The evolution of corporate web presence: A longitudinal study of large American companies," *International Journal of Information Management,* vol. 26, no. 4, pp. 313-325, 2006.

[2] C. &. R. S. Elliott, "Towards an understanding of corporate web identity," in *The Routledge Companion to Visual Organization*, 2014, pp. 273-288.

[3] Q. &. G. P. Yun, "The web mirrors value in the real world: comparing a firm's valuation with its web network position," *Computational and Mathematical Organization Theory,* vol. 21, no. 4, pp. 356-379, 2015.

[4] Neo4j, "Connected Components - Neo4j Documenation," 2018. [Online]. Available: https://neo4j.com/docs/graph-algorithms/current/algorithms/connected-components/.

[5] Neo4j, "Label Propogation - Neo4j Documentation," 2018. [Online]. Available: https://neo4j.com/docs/graph-algorithms/current/algorithms/label-propagation/.

[6] Neo4j, "The PageRank Algorithm - Neo4j Documentation," 2018. [Online]. Available: https://neo4j.com/docs/graph-algorithms/current/algorithms/page-rank/.

[7] Neo4j, "The Jaccard Similarity Algorithm - Neo4j Documentation," 2018. [Online]. Available: https://neo4j.com/docs/graph-algorithms/current/algorithms/similarity-jaccard/.

[8] Neo4j, "Overlap Simialrity - Neo4j Documenation," 2018. [Online]. Available: https://neo4j.com/docs/graph-algorithms/current/algorithms/similarity-overlap/.

[9] PwC, "Global top 100 companies by market capitalization," 2016. [Online]. Available: https://www.pwc.com/gr/en/publications/assets/global-top-100-companies-by-market-capitalisation.pdf.

[10] PwC, "Global 100 Software Leaders," 2016. [Online]. Available: https://www.pwc.com/gx/en/technology/publications/global-software-100-leaders/assets/global-100-software-leaders-2016.pdf.

[11] Forbes , "Forbes," 2016. [Online]. Available: https://www.forbes.com/sites/samanthasharf/2016/05/26/the-worlds-largest-tech-companies-2016-apple-bests-samsung-microsoft-and-alphabet/#3aabe951b661.

Appendix A - Code

Preprocess CSV's

```
json2csv -i companies.json -f id,update_time,domain,name,username -o companies.csv

json2csv -i links.json -f
id,update_time,home_name,home_domain,link_name,link_domain,username -o companies.csv
```

Load the Pre-Processed CSV's

```
CALL apoc.load.csv('companies.csv') yield lineNo, map, list
create (n:Companies) set n+=map;

call apoc.load.csv("links.csv") yield lineNo,map,list
match (a:Companies {domain:map.home_domain})
match (b:Companies {domain:map.link_domain})
call apoc.create.relationship(a,map.type,{},b) yield rel
return a,b,rel as relationships
```

Count the Number of Relationships

```
MATCH (n)-[r:partnership]->() RETURN COUNT(r)
MATCH (n)-[r:investment]->() RETURN COUNT(r)
MATCH (n)-[r:supplier]->() RETURN COUNT(r)
MATCH (n)-[r:customer]->() RETURN COUNT(r)
MATCH (n)-[r:competitor]->() RETURN COUNT(r)
```

Run Connected Components

All

```
CALL algo.unionFind.stream('Companies', '*', {})
YIELD nodeId,setId
RETURN count(distinct setId)
```

Per Relationship Type

```
CALL algo.unionFind.stream('Companies', 'partnership', {})
YIELD nodeId,setId
RETURN count(distinct setId)

CALL algo.unionFind.stream('Companies', 'investment', {})
YIELD nodeId,setId
RETURN count(distinct setId)

CALL algo.unionFind.stream('Companies', 'supplier', {})
YIELD nodeId,setId
RETURN count(distinct setId)

CALL algo.unionFind.stream('Companies', 'customer', {})
YIELD nodeId,setId
RETURN count(distinct setId)

CALL algo.unionFind.stream('Companies', 'competitor', {})
YIELD nodeId,setId
RETURN count(distinct setId)
```

Run Label Propagation

All

```
CALL algo.labelPropagation.stream("Companies", "*",{direction: "OUTGOING", iterations:
10})
YIELD nodeId, label
RETURN algo.getNodeById(nodeId).domain AS domain,label
Order by label desc

CALL algo.labelPropagation.stream("Companies", "*",{direction: "INCOMING", iterations:
10})
YIELD nodeId, label
RETURN algo.getNodeById(nodeId).domain AS domain,label
Order by label desc
```

Per Relationship Type

```
CALL algo.labelPropagation.stream("Companies", 'partnership',{direction: "OUTGOING",
iterations: 10})
YIELD nodeId, label
RETURN algo.getNodeById(nodeId).domain AS domain,label
Order by label desc

CALL algo.labelPropagation.stream("Companies", 'partnership',{direction: "INCOMING",
iterations: 10})
YIELD nodeId, label
RETURN algo.getNodeById(nodeId).domain AS domain,label
Order by label desc

CALL algo.labelPropagation.stream("Companies", 'investment',{direction: "OUTGOING",
iterations: 10})
YIELD nodeId, label
RETURN algo.getNodeById(nodeId).domain AS domain,label
Order by label desc

CALL algo.labelPropagation.stream("Companies", 'investment',{direction: "INCOMING",
iterations: 10})
YIELD nodeId, label
RETURN algo.getNodeById(nodeId).domain AS domain,label
Order by label desc

CALL algo.labelPropagation.stream("Companies", 'supplier',{direction: "OUTGOING",
iterations: 10})
YIELD nodeId, label
RETURN algo.getNodeById(nodeId).domain AS domain,label
Order by label desc

CALL algo.labelPropagation.stream("Companies", 'supplier',{direction: "INCOMING",
iterations: 10})
YIELD nodeId, label
RETURN algo.getNodeById(nodeId).domain AS domain,label
Order by label desc

CALL algo.labelPropagation.stream("Companies", 'customer',{direction: "OUTGOING",
iterations: 10})
YIELD nodeId, label
RETURN algo.getNodeById(nodeId).domain AS domain,label
Order by label desc
```

```
CALL algo.labelPropagation.stream("Companies", 'customer',{direction: "INCOMING",
iterations: 10})
YIELD nodeId, label
RETURN algo.getNodeById(nodeId).domain AS domain,label
Order by label desc

CALL algo.labelPropagation.stream("Companies", 'competitor',{direction: "OUTGOING",
iterations: 10})
YIELD nodeId, label
RETURN algo.getNodeById(nodeId).domain AS domain,label
Order by label desc

CALL algo.labelPropagation.stream("Companies", 'competitor',{direction: "INCOMING",
iterations: 10})
YIELD nodeId, label
RETURN algo.getNodeById(nodeId).domain AS domain,label
Order by label desc
```

Run Page Rank

Per Relationship Type

```
CALL algo.pageRank.stream('Companies', 'supplier', {iterations:20,
dampingFactor:0.85})
YIELD nodeId, score
RETURN algo.getNodeById(nodeId).domain AS page,score
ORDER BY score DESC

CALL algo.pageRank.stream('Companies', 'partnership', {iterations:20,
dampingFactor:0.85})
YIELD nodeId, score
RETURN algo.getNodeById(nodeId).domain AS page,score
ORDER BY score DESC

CALL algo.pageRank.stream('Companies', 'investment', {iterations:20,
dampingFactor:0.85})
YIELD nodeId, score
RETURN algo.getNodeById(nodeId).domain AS page,score
ORDER BY score DESC

CALL algo.pageRank.stream('Companies', 'customer', {iterations:20,
dampingFactor:0.85})
YIELD nodeId, score
RETURN algo.getNodeById(nodeId).domain AS page,score
ORDER BY score DESC

CALL algo.pageRank.stream('Companies', 'competitor', {iterations:20,
dampingFactor:0.85})
YIELD nodeId, score
RETURN algo.getNodeById(nodeId).domain AS page,score
ORDER BY score DESC
```

All

```
CALL algo.pageRank.stream('Companies', '*', {iterations:20,
dampingFactor:0.85,concurrency:8})
YIELD nodeId, score
RETURN algo.getNodeById(nodeId).domain AS page,score
ORDER BY score DESC
```

Run Jaccard and Overlap Similarity

Run Jaccard on Entire Graph, Filter for Top Companies

```
MATCH (a:Companies)-[r]-(b)
WITH {item:id(a), categories: collect(id(b))} as companyData
WITH collect(companyData) as data
CALL algo.similarity.jaccard.stream(data,{topK:1, similarityCutoff:
0.1,concurrency:8})
YIELD item1, item2, count1, count2, intersection, similarity
where
algo.getNodeById(item1).domain = "google.com" or
algo.getNodeById(item1).domain = "facebook.com" or
algo.getNodeById(item1).domain = "apple.com" or
algo.getNodeById(item1).domain = "ibm.com" or
algo.getNodeById(item1).domain = "amazon.com" or
algo.getNodeById(item1).domain = "microsoft.com" or
algo.getNodeById(item1).domain = "oracle.com"
RETURN algo.getNodeById(item1).domain AS home_domain, algo.getNodeById(item2).domain
AS link_domain, intersection, similarity
order by similarity desc
```

Run Jaccard only Between Top Companies

```
MATCH (a:Companies)-[r]-(b)
WHERE
a.domain ="microsoft.com" or
a.domain="google.com" or
a.domain="ibm.com" or
a.domain="facebook.com" or
a.domain="oracle.com" or
a.domain="amazon.com" or
a.domain="apple.com"
WITH {item:id(a), categories: collect(id(b))} as companyData
WITH collect(companyData) as data
CALL algo.similarity.jaccard.stream(data,{topK:1, similarityCutoff:
0.1,concurrency:8})
YIELD item1, item2, count1, count2, intersection, similarity
RETURN algo.getNodeById(item1).domain AS home_domain, algo.getNodeById(item2).domain
AS link_domain, intersection, similarity
Order by similarity DESC
```

Run Overlap on Entire Graph, Filter for Top Companies

```
MATCH (a:Companies)-[r]-(b)
WITH {item:id(a), categories: collect(id(b))} as companyData
WITH collect(companyData) as data
CALL algo.similarity.overlap.stream(data,{topK:1, similarityCutoff:
0.1,concurrency:8})
YIELD item1, item2, count1, count2, intersection, similarity
where
algo.getNodeById(item1).domain = "google.com" or
algo.getNodeById(item1).domain = "facebook.com" or
algo.getNodeById(item1).domain = "apple.com" or
algo.getNodeById(item1).domain = "ibm.com" or
algo.getNodeById(item1).domain = "amazon.com" or
algo.getNodeById(item1).domain = "microsoft.com" or
algo.getNodeById(item1).domain = "oracle.com"
RETURN algo.getNodeById(item1).domain AS home_domain, algo.getNodeById(item2).domain
AS link_domain, intersection, similarity
order by similarity desc
```

Run Overlap only Between Top Companies

```
MATCH (a:Companies)-[r]-(b)
WHERE
a.domain="google.com" or
a.domain="microsoft.com" or
a.domain="ibm.com" or
a.domain="facebook.com" or
a.domain="oracle.com" or
a.domain="amazon.com" or
a.domain="apple.com"
WITH {item:id(a), categories: collect(id(b))} as companyData
WITH collect(companyData) as data
CALL algo.similarity.overlap.stream(data,{topK:1, similarityCutoff:
0.1,concurrency:8})
YIELD item1, item2, count1, count2, intersection, similarity
RETURN algo.getNodeById(item1).domain AS home_domain, algo.getNodeById(item2).domain
AS link_domain, intersection, similarity
Order by similarity DESC
```