We have the following graph structure given: $G = (V, E)$ where the set of vertices $V = M \cup S$ is the disjoint union of the set of mashups M and the set of web services S of our dataset. The set of edges E is defined as: $E = \{(s, m) \subseteq S \times M : \text{Mashup m uses the API of service s.}\}$

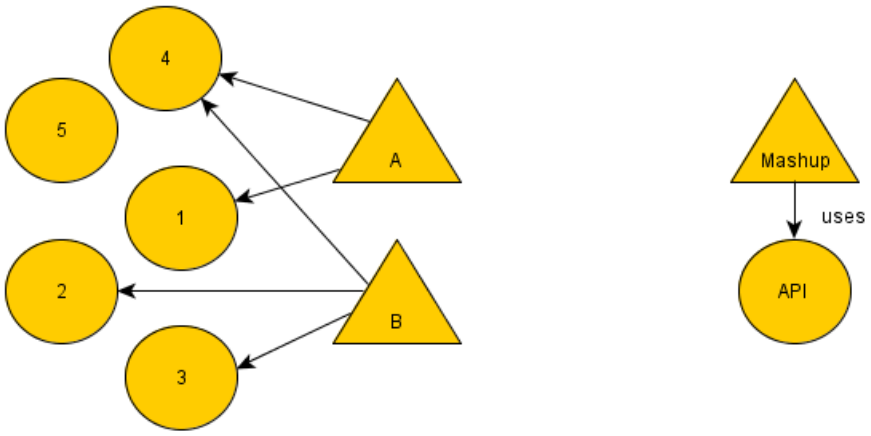

Figure 1: Illustration of the Mashup-API-Graph

We now simplify this graph a bit. We now have a *bipartite, disconnected and undirected* graph G'.

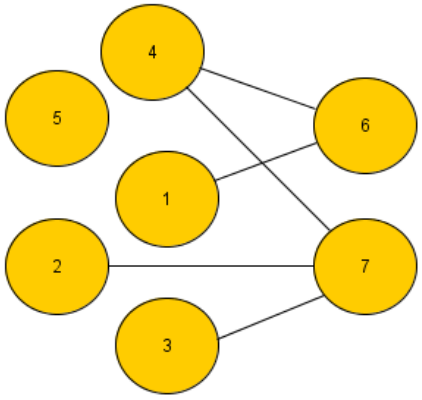

Figure 2: Illustration of the simplified Mashup-API-Graph

The calculations of the centralities are done on this simplified graph. The distinction between APIs and mashups is not relevant for it. We only select

APIs from the triple store and the centralities of their node representatives are used.

1 Degree Centrality

The number of incident edges. (The number of mashups that use an API)

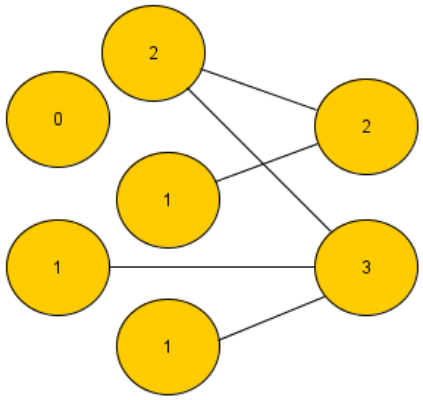

Figure 3: Illustration of Degree Centrality

2 Betweenness Centrality

Let $dist(u, v)$ denote the *distance*, i.e. the length of a *shortest path*, between the vertices u and v and SP_{uv} the set of different shortest paths between u and v while $\sigma_{uv} = |SP_{uv}|$ denotes the number of those different shortest paths. For each $k \in V, k \neq u, k \neq v$ let $SP_{uv}(k)$ - and $\sigma_{uv}(k) = |SP_{uv}(k)|$ accordingly - denote the set of *shortest paths* between u and v that contain the vertex k.

Definition 1. Betweenness Centrality *of a vertex $v \in V$ is the sum of the fractions of shortest paths between $s \neq t \in V$ containing v to the total amount of shortest paths between s and t.*

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}, v \in V \tag{1}$$

Using Brandes' algorithm [1]. Still very slow to calculate - therefore precalculated.

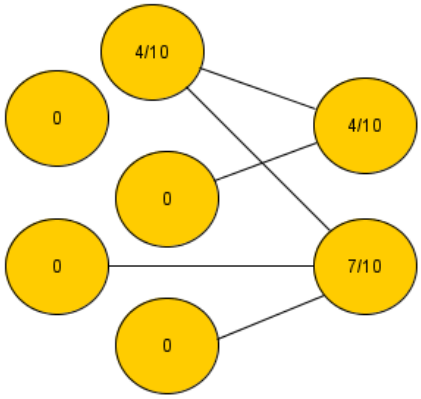

Figure 4: Illustration of Betweenness Centrality

3 Closeness Centrality

Definition 2. Closeness Centrality *with [2]'s modification and Brandes' algorithm [1], too. Therefore calculated before as well.*

$$C_C(v) = \sum_{t \in V} \frac{1}{dist(v,t)}, v \in V \tag{2}$$

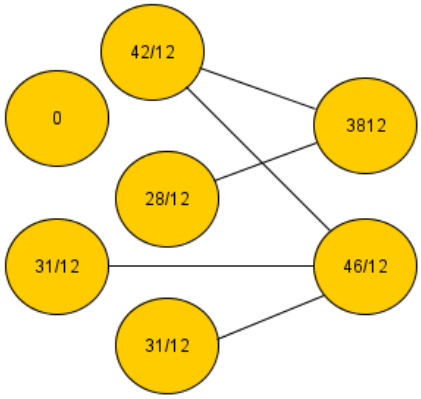

Figure 5: Illustration of Closeness Centrality

4 Eigenvector Centrality

- calculated "live"
- uses Power Iteration
- http://www.ams.org/samplings/feature-column/fcarc-pagerank

5 Combining Ranking Functions

Please consider the following example: We have three web services in our subset, s_A, s_B and s_C, as well as two elementary ranking functions f_1 and f_2 which produce the following ranking scores. The function f_1 ranks service s_A as the most relevant one with a score of 10, s_B in second place scoring 5 and s_C in third place with a score of 1 whereas function f_2 places s_B (score of 8) first, s_C (score: 3) second and s_A last with a score of 1. In order to create a new ranking, we can now combine f_1 with f_2 with $\lambda_1 = \lambda_2 = 0.5$. The resulting ranking of our linear combination $\mathcal{F}$ would be the following. Service s_B scores $0.5*8+0.5*5 = 6.5$, s_A scores $0.5*10+0.5*1 = 5.5$ and s_C scores $0.5*1+0.5*3 = 2.0$. Table 1 illustrates this example. Depending on the criteria for the evaluation one can now decide whether the new ranking represents an improvement or not.

Table 1: Illustration of the example scenario for a linear combination of ranking functions

Pos	f_1 (score)	f_2 (score)	$\mathcal{F}$ (score)
1	s_A (10)	s_B (8)	s_B (6.5)
2	s_B (5)	s_C (3)	s_A (5.5)
3	s_C (1)	s_A (1)	s_C (2)

References

[1] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[2] T. Opsahl, F. Agneessens, and J. Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks 32 (3)*, pages 245–251, 2010.