Ranking of Legal Cases

EECS 718 Final Project

By Michael Dreiling (2157982)
Under direction of Professor Doug Niehaus

May 17, 2011

1 Abstract

The web is a directed graph where webpages interlink amongst each other. Google, and Jon Kleinberg, used this linked nature to determine the rank of a webpage. Likewise, the use of judges citing the rulings of previous cases within their arguments creates a directed graph of the "legal web". By using this graph of the legal web and its subgraphs, two rankings are proposed. The first, a query-independent ranking, creates a hybrid of Google's PageRank and Jon Kleinberg's HITS algorithms. The second, a query-dependent ranking, attempts to rank each case based on the connectivity, or spectral gap, of a particular subgraph of this legal web.

Contents

2 Introduction

Ranking algorithms are nothing new. Within the past decade alone, with Google pioneering the way, everything from movies (Netflix), books (Amazon), and music (Pandora), to drugs, locations (SkyHook), and even people (UserRank) have been ranked. In each case, the most successful algorithms (that had the intention of turning into a company) have all thrived into incredibly successful businesses. The public realm of legal cases, however, has been relatively untouched by this wave of ranking despite its arguably more important need for such a tool. The United States court system, in which court opinions interpret the law and set a precedent for future decisions, is a system where precedent presides. Lower courts must follow the law as it is interpreted by higher courts, and any court generally tries to follow its own precedent from previous decisions. Thus, the persuasive authority of similar case's opinions can be used by a lawyer to build an argument for a case s/he is researching. A method that can rank legal cases based on a particular subject matter is then, in many ways, priceless for lawyers performing case research.

Several similarities can be found between the ranking of legal cases and the traditional ranking of webpages. Google determines the rank of a webpage based on the rank, or importance, of the pages it links to. Analogous to how the importance of a person is not just the importance of that person alone, but also of the sum of importance and influence of his/her contacts. Legal cases can be thought of in a similar manner, where the "rank" of a legal case is not only the influence that case holds but the sum of influences of the cases that cite it. These types of rankings are global, that is, they take into account the entirety of the web, all people on earth, or every case ever ruled. This is, in search theory, typically referred to as **query-independent**. The "rank" of an object is known well before a person ever searches for it. While this type of ranking can be prudent for legal cases, what is more desirable is a **query-dependent** ranking. That is, a ranking based on the subject matter that initiated the search. The overall "popularity" of a case may be interesting, but the influence a case has had specifically on, for example, contracts, is infinitely more valuable to a lawyer. So how are these types of rankings performed? And is it possible and or better to include some kind of hybrid of query-independent and query-dependent rankings? These are the questions this paper will explore.

3 What is a legal case?

Before diving into the ranking algorithms, it's worth understanding exactly what a legal case (or legal opinion, if you will) is and how it relates to common law so legal-specific enhancements can be applied to the ranking.

After there is a ruling on a case, the judge/judges involved write an opinion outlining the facts, applicable law, and the rationale supporting the decision. In essence, the judge writes why he ruled the way he did. These statements may be as brief as a couple of sentences or as long as several hundred pages. If there is more than one judge, three situations can occur. (1) A judge may agree completely with the original written statement and simply sign it. (2) A judge may agree with the final verdict, but for differing reasons. This is referred to as a concurring opinion and the judge/judges may write a second opinion on the case. And (3), a judge may disagree with the final verdict. This is referred to as a dissenting opinion and again the judge/judges may write an opinion outlining their point of view. When the majority of judges agree with the original opinion, that opinion is said to be the *majority opinion* and consequently carries a significant amount of precedental value. When the original opinion does not contain the majority, the opinion is considered a *plurality opinion*. A plurality may occur where, for example, four of nine justices join one opinion, two others write concurrences, and three write dissents. While this opinion is still important, it carries less precedental value than a majority opinion. These opinions (the majority and plurality opinions) are what will, in the end, be ranked.:

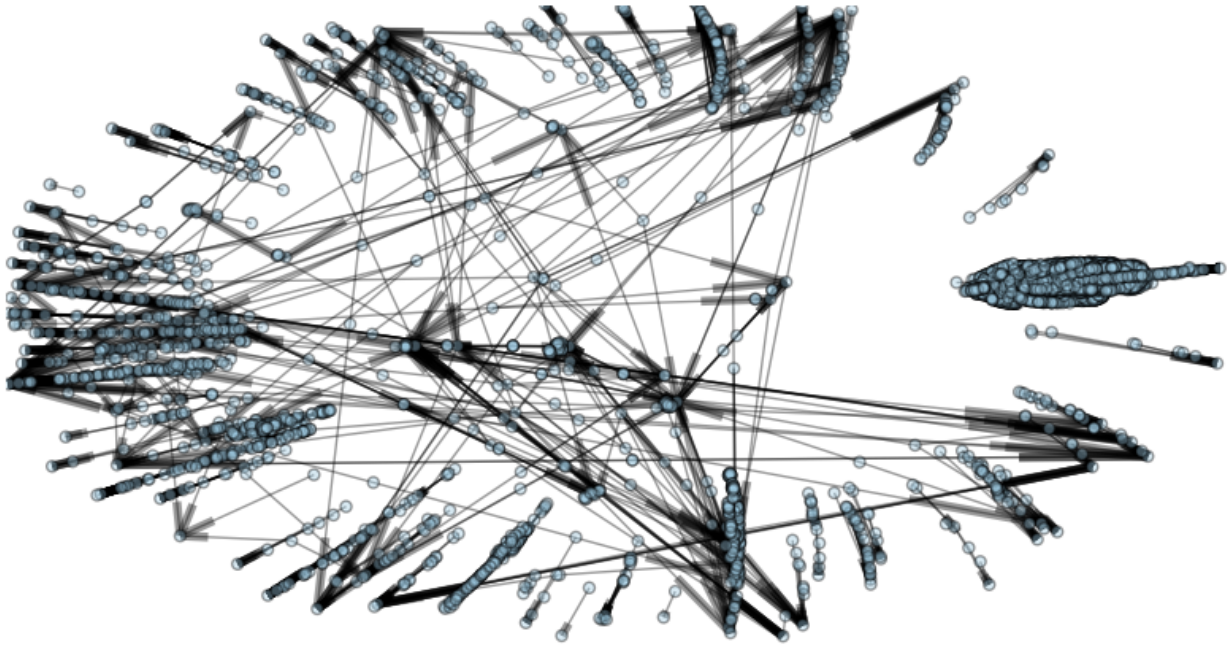
```
There are other, less frequently encountered (but perhaps more important),
types of opinions, such as "En Banc". These types are not considered in
this paper.
```

The legal doctrine of *stare decisis*, which means "to stand by that which is decided", typically dictates that judges are obliged to respect the precedents established by prior decisions. As such, judges (and

lawyers) use this principle to build an argument for why a case should be ruled a particular way. Once an opinion has been published, judges using said opinion will cite that case in their statement. Over time a case may, therefore, acquire a significant amount of opinions citing it.

Higher court rulings overrule (or carry an infinite amount more precedent) over lower courts. The federal court structure is split into three main categories. At the top is the Supreme Court. Next are the court of appeals. These include the 12 regional "circuit" courts, including the D.C. Circuit, and carry no original jurisdiction. The lowest level contains the district courts. These include the 94 judicial districts spread throughout the 50 states and territories and include original jurisdiction. Most cases start at the district court level. Once a district court judge issues a ruling, a case can proceed to the circuit court level, or even all the way up to the Supreme Court. However, cases can move down the structure as well. A complex case may go back and forth among the different levels one or more times.

4 A Query-Independent Ranking



4.1 Ranking by Influence

The practice of judges citing previously ruled cases in their opinions creates an enormous directed graph where each node (a legal opinion) contains both **outlinks** (edges pointing to the opinions the judges cited in their opinion) and, with time, **inlinks** as other judges cite the ruling of that case in their own opinions.

Google's PageRank thesis says that a "webpage is important if it is pointed to by other important pages". Similarly, we can define that **a legal opinion is important if it is pointed to (cited) by other important cases**. From this, a simple ranking for a case C_i, denoted $r(C_i)$, can be obtained by summing the ranks of all case C_i's inlinks

$$r(C_i) = \sum_{C_j \in I_{C_i}} r(C_j)$$

where I_{C_i} is the set of all cases inlinking to case C_i. With this equations's simplicity, however, comes several weaknesses. First, not every inlink is inherently "equal" (despite their different ranks). Legal opinions often cite a case in varying extensiveness. One opinion may, for example, simply mention the case while another may discuss it fully in its argument. This presents a situation where a highly ranked case, which simply mentions a relatively unimportant case, can skew a case's ranking. The two main legal research providers, Westlaw and Lexis/Nexis, provide a four level classification system that depicts the

extent at which a case discusses a cited case. For clarity, Westlaw's is presented below:

Class Number	Classification Description
1	A brief reference to the cited case, usually in a string citation.
2	Some discussion of the cited case, usually less than a paragraph.
3	Substantial discussion of the cited case, usually more than a paragraph but less than a printed page.
4	Discussion of the cited case, usually more than a printed page of text.

This classification system can be used to temper the rank for inlinks as they are being summed. By introducing a coefficient, ω, into the summation, the inlink's weight can be more accurately defined by taking the class the inlink is in (one of the four above) divided by the number of classes (4 in this case). An inlink with a class of *1*, for example, would then have an omega value of 0.25. To clarify further, a case that simply cites the case being ranked without discussing it in any way (class 1) would then only have 25% of its full rank added. Likewise, a case that extensively discusses the case being cited (class 4) would, rightfully, have 100% of its full rank added. Because these values are unique for each inlink, additional notation is introduced to denote which case is being cited and which case is citing it. $\omega_{C_j \to C_i}$ denotes the omega value for case C_j citing case C_i. The equation then becomes:

$$r(C_i) = \sum_{C_j \in I_{C_i}} \omega_{C_j \to C_i} \cdot r(C_j)$$

Google's PageRank, in an attempt to devalue the weight of pages who spam links, also tempers the rank of inlinks, but by the number of outlinks for that page (not by extensiveness). This equation, taking into account the differing methods for tempering the weight of inlinks, is equivalent to Google's originally published PageRank summation equation.

A second problem, not solved in this new equation and not related to the ranking of webpages, is that this equation naturally favors older cases. The longer a case has been published, the more time it has had to gain citing cases. I've pondered whether or not this is truly a significant problem, as an older case should, after all, carry more precedental value. Original cases that have survived the onslaught of Common Law and still provide an authoritative answer for a particular matter should, in my mind, be ranked first. Regardless, my solution to solving this situation is by taking into account the case's outlinks in the same way as the case's inlinks. This doesn't hurt the rankings in anyway, it only enhances them. Informally, this can be described by saying a case is important if it is cited by other important cases **and also cites other important cases**. Formally, the equation now becomes:

$$r(C_i) = \sum_{C_j \in I_{C_i}} \omega_{C_j \to C_i} \cdot r(C_j) + \sum_{C_k \in O_{C_i}} \omega_{C_i \to C_k} \cdot r(C_k)$$

where O_{C_i} is the set of all cases outlinking from case C_i. The first summation for recently published (young) cases will, obviously, be very low. The second summation, however, equally treats all published cases regardless of the age of the case (except for, of course, the first cases ever published).

At this point, the equation, $r(C_i)$, begins to look like a hybrid of Google's PageRank algorithm and a query-independent version of Jon Kleinberg's HITS algorithm (where, in this case, hub and authority scores are added together). This adds another advantage to the ranking on top of deterring the aging problem: it provides the user flexibility. At any point one of the summations can be ignored in the sorting of search results, presenting, according to HITS, good *authorities* or good *hubs*. In this sense, a good authority would be a case in which several other important cases *cite* that case in their opinions. A good hub, on the other hand, would be a case that *cited* several other important cases in its opinion. An important case, as is defined above, is both of a good authority *and* a good hub (this definition, despite only subtle changes, produces completely different results compared to HITS).

4.2 Calculating the Ranking

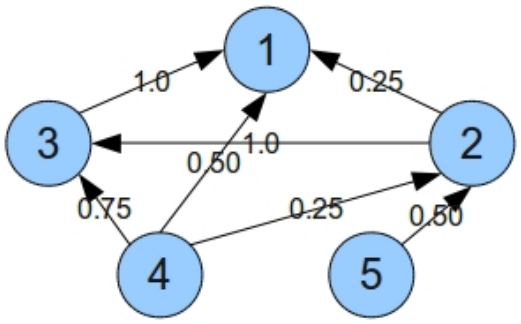
This final equation (along with the others), as you may have noticed, is circular. The rank of a case depends on the ranks of other cases, and those other case's ranks depend on the rank of yet more cases. How is this seemingly endless loop solved? To my benefit, the vast amount of research on both PageRank and HITS deals exactly with this.

In summary, all the cases are given an initial ranking, say $1/n$, where *n* is the number of cases currently indexed. Then, by using these initial values, the ranks for all cases can be calculated, giving a (very) rough estimate to their true values. These calculations are then repeated using these newly calculated rankings, giving yet another rough estimate. Through more and more iterations of this process, the rankings begin to converge to their "true" values.

This process can be compactly described by representing the calculations in matrix form, where the rank of all cases at a current iteration, k, are contained in a column vector, r_k, the omega coefficients are contained in an *n x n* matrix, Ω, and the rankings of the previous iteration are contained in another column vector r_{k-1}.

$$r_0 = e/n$$
$$r_k = \Omega^T r_{k-1} + \Omega r_{k-1}$$

The structure of Omega is, essentially, the adjacency matrix for the directed graph depicting all legal cases currently indexed, where, instead of a value of 1 for a link from case i to case j, the value is the link's omega value (Or, in other words, the Omega matrix lists the *weight* of all directed edges for the graph where the weight of an edge is given by its omega value).



$$\Omega = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0 & 1.0 & 0 & 0 \\ 1.0 & 0 & 0 & 0 & 0 \\ 0.5 & 0.25 & 0.75 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \end{pmatrix}$$

Because it may be useful to ignore either one of the original summations to acquire the authority and hub rankings, it's more prudent to define each calculation separately. The column vector based on the rank of inlinks is then $x_k = \Omega^T r_{k-1}$. Likewise, the column vector based on the rank of outlinks is $y_k = \Omega r_{k-1}$. This is more or less the **iterative power method** for computing the **dominant eigenvector** for Ω, and Ω^T. The final vector of rankings at some iteration *k* is $r_k = x_k + y_k$. Results for the first two iterations (along with the initial values), using the example graph above, are shown below:

Case	r_0	x_1	y_1	$r_1 (x_1 + y_1)$	x_2	y_2	$r_2 (x_2 + y_2)$
1	0.20	0.35	0.00	**0.35**	0.4125	0.00	**0.4125**
2	0.20	0.15	0.25	**0.40**	0.125	0.6375	**0.7625**
3	0.20	0.35	0.20	**0.55**	0.475	0.35	**0.825**
4	0.20	0.00	0.30	**0.30**	0.00	0.6875	**0.6875**
5	0.20	0.00	0.10	**0.10**	0.00	0.20	**0.20**

Notice that case 4, which in this example depicts possibly a young case (as it has no inlinks), is ranked 3rd overall. Case 5 is, as should be expected, ranked the lowest. Also worth noticing is that if, for whatever reason, a researcher wanted his/her results sorted only by cases that were *cited* by other important cases the inlink vector, x, would correctly have case 1 and 3 listed highest. The alternative can be said for the outlink vector, y, where after sorting based on cases that *cite* other important cases, would

correctly have cases 2 and 4 listed highest.

There's still two enhancements that can be made to this calculation. First, as the equation currently is, rankings of 0 (and consequently, ties at 0), are possible and also likely for the inlink ranking (see case 4 and 5). Second, there's no guarantee that these calculations will eventually converge, or, even if they will, converge in a reasonable amount of iterations.

Both of these problems are, in fact, related. Ties at 0 make it impossible to guarantee convergence. To solve this, modifications to the Omega matrix, Ω, the inlink ranking, x_k, and the outlink ranking, y_k, can be made. By multiplying the Omega matrix by a scaling factor, ξ, where $0<\xi<1$, and adding on a constant column vector based on that scaling factor, $(1-\xi)/n \cdot e$, the *x* and *y* vectors are now positive (contain no zero entries) and the Omega matrix is now irreducible. Then by normalizing the calculations using a vector-1 normal, denoted $||r_k||_1$, after each calculation (making the column vector, r_k, sum to 1) the power method is guaranteed to converge in a finite number of iterations. The column vectors *x*, *y*, and *r* are now defined as:

$$x_k = \xi \Omega^T r_{k-1} + (1-\xi)/n \cdot e$$
$$y_k = \xi \Omega r_{k-1} + (1-\xi)/n \cdot e$$
$$r_k = x_k + y_k$$
$$r_k = r_k / ||r_k||_1$$

The calculations for x_k, and y_k, are actually calculating what's called the **Perron vector**: a unique, normalized (well, if I normalized *x* and *y* -- I'm still split about this decision), positive dominant eigenvector. Using the initial 5 case example again, the new ranks for each case, using $\xi = 0.95$, are presented below:

Case	r_0	x_1	y_1	r_1	x_2	y_2	r_2
1	0.20	0.3425	0.0100	**0.2055**	0.4503	0.0100	**0.2306**
2	0.20	0.1525	0.2475	**0.2322**	0.0841	0.3593	**0.2222**
3	0.20	0.3425	0.2000	**0.3163**	0.3582	0.2052	**0.2823**
4	0.20	0.0100	0.2950	**0.1778**	0.0100	0.3884	**0.1996**
5	0.20	0.0100	0.1050	**0.0671**	0.0100	0.1208	**0.0655**

Notice that there are no ranks equal to 0, all the *r* vectors sum to 1 (pending rounding, actual implementations should go to far more than 4 decimal places), and that the rankings appear to be converging faster. The overall rankings after the first iteration match the rankings from the unmodified equations. By the second iteration, however, the rankings more accurately depict the "percentage" of rank each case holds.

The *x* and *y* vectors don't sum to 1 because they are not normalized after each iteration. I tested this in numerous different situations, and in each case the overall ranks were relatively unchanged. If I were to normalize both of those vectors, there would be an increased cost of *2n* additional addition operations per iteration, despite no change in rankings. It's because of this that I decided only to normalize the final ranking vector *r*.

```
Additional research, if I was given the time (this would have been a GREAT
semester-long project), would have explored basing the starting values on
majority/plurality opinions in conjunction with the level of court the
case was heard. This would give supreme court cases, for example, a
"head start" in the rankings compared to district court cases. Note that
whatever method used, the starting values should sum to one.
```

4.2.1 The Final Algorithm

Putting everything together, the final algorithm for computing my query-independent ranking for a set of legal cases is:

1. Initialize $r_0 = e/n$, where *e* is a column vector of all ones.
2. Until convergence do:
   $x_k = \xi \Omega^T r_{k-1} + (1-\xi)/n \cdot e$
   $y_k = \xi \Omega r_{k-1} + (1-\xi)/n \cdot e$
   $r_k = x_k + y_k$
   $r_k = r_k / ||r_k||_1$
   $k = k+1$
3. Set the authority (inlink ranking) vector $x = x_k$, and the hub vector (outlink ranking) vector $y = y_k$. There is no need to normalize these.
4. Set the final ranking vector $r = r_k$.

Research on HITS using the same modifications I've used have found that only 10 to 15 iterations are needed until convergence. A python implementation of this algorithm is presented in Appendix A.

4.3 Complexity Analysis

The bottleneck of this algorithm is, of course, the two matrix-vector multiplications - one for each vector x, and y. The complexity of these boils down to the number of non-zero entries in the weighted adjacency matrix, Ω. Additionally, per every iteration, $4n$additions are performed (1 for each ranking vector, 1 for $r_k = x_k + y_k$, and 1 for the vector-1 normal). From this, the complexity is (more or less):

$$Cost(r) = k * (2nnz(\Omega) + 4n)$$

where the term, $nnz(\Omega)$, denotes the number of non-zero entries in the matrix Ω. This is roughly twice the cost of Google's PageRank, which makes sense, as I'm computing two ranking vectors as opposed to one.

While the nature of the web allows the number of inlinks and outlinks for webpages to be dynamic, legal opinions have the benefit of being rather static. The outlinks of a particular case will never change once published (but the ranks for those outlinks will, over time, change), and the growth of the inlinks for the vast majority of opinions is significantly slower than that of the web. So, even though the calculations for this query-independent algorithm may be slower than a direct PageRank or HITS equivalent, the calculations don't need to be performed as often. To think about it another way, since the ranks of legal cases change several orders of magnitude slower than webpages, it is acceptable to use an algorithm that may (or may not, since there are significantly fewer published legal opinions then there are webpages, I would think) be slower and update the ranks less often.

4.4 Sample Results

The following is my algorithm applied to a database of several thousand legal cases (3898 to be exact). Only the top 30 cases are shown. Due to the nature of this database and the cases contained within it (all cases are derived from a single root case), the rankings are not as accurate as they could be otherwise. As the database grows and becomes more and more complete, the rankings will also become more accurate.

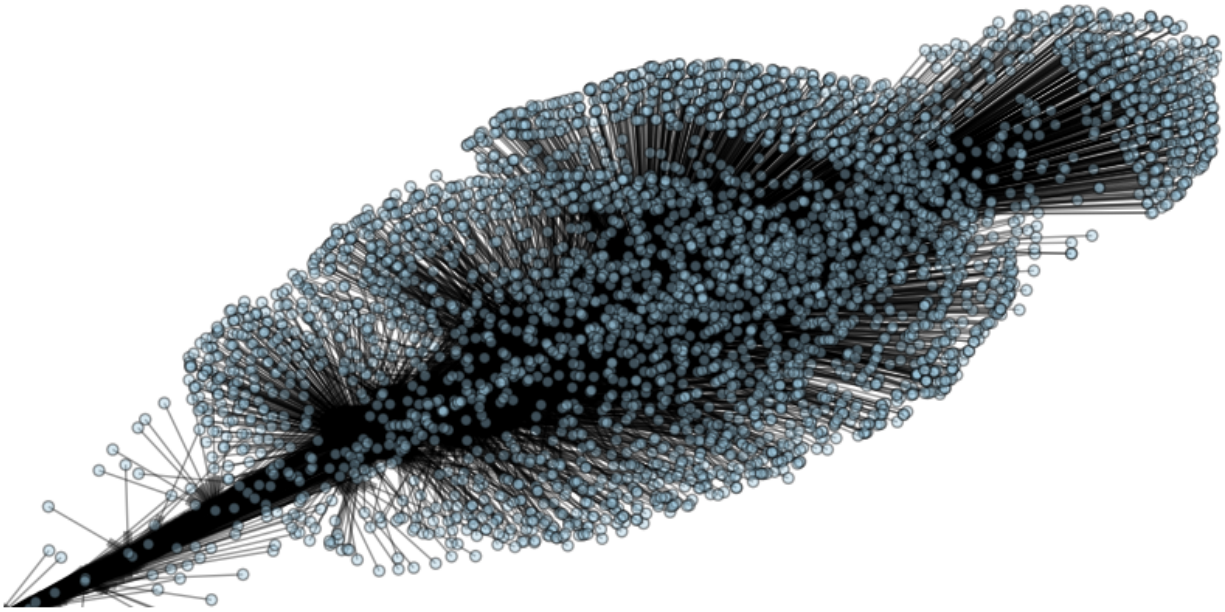
Interestingly, the root case in which all other cases were pulled was actually ranked 6 overall (Talton v. Mayes). This case was of interest to the (current?) dean of the law school, and as such most cases are related in some way to Indian tribes. My algorithm performed rather well, as the first four cases were all Supreme Court cases (as well as 6, 8, and 9), and cases ranked 5, 15, and 16 were all ruled recently (2009, 2005, 2005, respectively), showcasing its immunity to favoring old cases.

R	$r(C)$	Case
1	0.0442748	Santa Clara Pueblo v. Martinez, 436 U.S. 49 (1978).
2	0.0142782	United States v. Wheeler, 435 U.S. 313 (1978).
3	0.0093526	Merrion v. Jicarilla Apache Tribe, 455 U.S. 130 (1982).
4	0.0055911	Oliphant v. Suquamish Indian Tribe, 435 U.S. 191 (1978).
5	0.0050140	Acosta-Vigil v. Delorme-Gaines, 672 F. Supp. 2d 1194, (D.N.M. 2009).
6	0.0048337	Talton v. Mayes, 163 U.S. 376, (1896).
7	0.0044495	Ordinance 59 Ass'n v. Babbitt, 970 F. Supp. 914, (D. Wyo. 1997).
8	0.0042359	Duro v. Reina, 495 U.S. 676 (1990).
9	0.0039639	Nevada v. Hicks, 533 U.S. 353 (2001).
10	0.0035216	Martinez v. Southern Ute Tribe, 249 F.2d 915 (10th Cir. Colo. 1957).
11	0.0035032	State ex rel. Railroad Comm'rs v. Atlantic C. L. R. Co., 60 Fla. 465 (1910).
12	0.0032571	Wright v. Melbert Prairie Chicken, 1998 SD 46 (1998).
13	0.0032106	Government of Virgin Islands v. Parrott, 476 F.2d 1058 (1973).
14	0.0031179	Atkinson Trading Co. v. Shirley, 210 F.3d 1247 (10th Cir. N.M. 2000).
15	0.0030894	Merrill Lynch Bus. Fin. Servs. v. Performance Mach Sys. U.S.A., 2005 U.S. Dist. LEXIS 7309 (S.D. Fla. Mar. 4, 2005).
16	0.0029608	Lewis v. Norton, 424 F.3d 959 (9th Cir. Cal. 2005).
17	0.0029064	Dodge v. Nakai, 298 F. Supp. 17 (D. Ariz. 1968).
18	0.0028709	St. Louis , I. M. & S. R. Co. v. Davis, 132 F. 629 (C.C.D. Ark. 1904).
19	0.0027238	Native Am. Mohegans v. United States, 184 F. Supp. 2d 198 (D. Conn. 2002).
20	0.0027130	United States v. One 1973 Buick Riviera Auto., 560 F.2d 897 (8th Cir. Ark. 1977).
21	0.0026641	Batten v. United States, 2005 U.S. Dist. LEXIS 45825 (N.D. Ohio Mar. 1, 2005).
22	0.0026639	State ex rel. Postal Telegraph-Cable Co. v. Wells, 96 Fla. 591 (1928).
23	0.0026541	Senate Select Committee v. Nixon, 366 F. Supp. 51 (D.D.C. 1973).
24	0.0026410	Schmuck v. United States, 489 U.S. 705 (1989).
25	0.0026148	United States v. Mitchell, 463 U.S. 206 (1983).
26	0.0026053	Oliphant v. Schlie, 544 F.2d 1007, 1976 U.S. App. LEXIS 7444 (9th Cir. Wash. 1976).
27	0.0025777	Trans-Canada v. Muckleshoot Indian Tribe, 634 F.2d 474 (9th Cir. Wash. 1980).
28	0.0025753	Wakaksan v. United States, 367 F.2d 639 (8th Cir. N.D. 1966).
29	0.0025461	Dry v. United States, 235 F.3d 1249 (10th Cir. Okla. 2000).
30	0.0025216	Wetsit v. Stafne, 44 F.3d 823 (9th Cir. Mont. 1995).

To rank this database, the only thing needed was to create the Omega, Ω, matrix. This matrix, as described in the previous sections, is an *n x n* weighted adjacency matrix, where the weight is given by the link's omega, ω, value. It's important to note that the database has no data structure or distinct storage for a case's outlinks. This, however, is not a problem, as the outlinks can be (and are) derived from the inlinks. Code for creating this Omega matrix is presented in Appendix A.

The combined building of the Omega matrix, pushing that matrix through the ranking algorithm, and then sorting the results took well less than a minute on my person laptop.

5 A Query-Dependent Ranking



5.1 Ranking by Relevance

```
The following section was my attempt at creating a query-dependent
ranking. I have since determined, at least initially with the given
time I had, that what I was trying to do is either infeasible or
impractical for ranking. The section is included for completeness.
```

Ranking by similarity presents several problems that are not easily overcome. When someone speaks about similarity, what they are usually implying is *relevant* similarity. Legal cases, as with almost anything, can be similar but in completely irrelevant ways. My knowledge of legal history is next to none, so I have no real-world example of this relating directly to legal cases, but a good analogy to ethics can be made. When determining what ethical action to make in a certain situation, often relating what is "ethically correct" in a similar situation is the best course to make. If someone sees a large man carrying a baseball bat at night at an inconspicuous location, should that person notify authorities? Well, consider if someone sees a large man carrying a baseball bat at night by a baseball stadium, do they contact authorities? Probably not. Both cases are similar in that they both involve a large man carrying a baseball bat, but are similar in completely irrelevant ways. The fine line between relevance and similarity (and the two together) is the main driving point behind query-dependent rankings. The query-independent ranking in section 4 compares every case to *every* other case, regardless of the topic that a user searched for. A query-dependent ranking, on the other hand, compares only the cases that were initially returned from a text search. From this perspective, it is much easier to then rank this very small subset based, not on influence or popularity, but on relevant similarity to the search term.

The only concern for query-dependent rankings, and indeed the biggest drawback, is that whatever calculations are required, they are required for *every* search performed. As such, whatever method developed, it must be efficient enough to rank the results in a small amount of time as to not annoy the user or keep the user waiting.

My approach was an attempt at using the spectral gap on some subgraph of each case's inlinks and outlinks. The spectral gap (of the subgraph's adjacency matrix) is the difference between the largest eigenvalue μ_0, and the second largest eigenvalue μ_1. If the difference is large, the graph is well-connected. If the difference is small, the graph is increasingly random or contains "randomness". The idea was to modify the subgraphs of each case's "citation relationship graph" to only include links which

were deemed relevantly similar to the search term, and then calculate the difference based on that subgraph. The cases would then be ranked based on which cases' graphs were "more connected".

The "citation relationship graph" is essentially just the graph of the root case along with its inlinks and outlinks. These original citation relationship graphs can be incredibly dense and interconnected for popular, well-cited cases. By removing any edges (but not the nodes) deemed irrelevant to the search term, we can gage how strong the root case, in terms of all its cited and citing cases, pertains to a particular subject matter. If the graph, after being modified, is still strongly connected, then it obviously both cited and was cited by cases that were also relevant to the search term. If it is not strongly connected then it means, more or less, that the case both cited and was cited by other cases along some other subject line.

This method didn't work for two reasons. First, the computations took about 50 milliseconds to perform for a single case. Multiply that for a search that returns 100 cases and it will take at least 5 seconds to display the results to the user (plus the time for index lookups and sorting). Second, the values I received weren't very distinctive. Some graphs, as seen above, have an enormous amount of inlinks and outlinks. Other graphs have next to none. The spectral gap values needed to be normalized some way to accurately depict each case's "connectedness" compared to others, despite the number of edges. I was unable to come up with an adequate normalizing method, either do to some lack of my knowledge on graph theory or do to the vast amount of time it has been since I've used linear algebra in any way, shape, or form. I was unable to accurately describe my problem to find relevant information to research and solve it. The lack of time given for this project (as well as my habit of being overambitious for interesting projects) also contributed to me being unable to solve this problem.

6 From Ranking to Searching

While applying these rankings in a useful manner is not the subject of this paper, (and, consequently, has no code presented), I believe it's prudent to at least expand on the idea of how these ranking algorithms could be used in searching.

6.1 Indexing

First and foremost, the database of legal opinions needs to be indexed so text search can be performed. Indexing is the process of storing a mapping of content to its locations.

6.2 Text Search

Once indexed, a user's search query is then used to retrieve a set of relevant pages based on lookups into the index. As such, the ability to accurately process a natural language query handling both synonymy and polysemy is critically important for useful searching. Synonymy is where multiple words have the same meaning, such as the age old example of car and automobile. Polysemy is where a single word contains multiple meanings, such as the term bank - which could mean a financial institution, curve in the road, etc. There are several open-source options for accomplishing this task at a more-than-adequate level, such as Apache Lucene.

6.3 Sorting by Rank

Once this set of relevant pages is retrieved, they are ranked (possibly using the algorithms presented in this paper) and then ordered from most relevant to least relevant based on this ranking. Often times the indexing system provides its own "content ranking" based purely on the text search. In this case the content ranking can be ignored or the two rankings can be added together and the set can be sorted based on that sum. After being sorted, the resulting set is then displayed to the user from most relevant to least relevant.

For query-independent rankings, the ranks are already known and no further calculations need to be performed. The main drawback for query-dependent rankings is that for every search additional calculations need to be performed, thereby slightly delaying the results to the user.

7 Conclusions

My original thought on the ranking of legal cases was that a query-independent ranking would be, in many ways, inadequate. What I've come to realize is the complete opposite. The notion that **a case is important if it both cites other important cases and is cited by other important cases** is incredibly natural, and, indeed, the subtleties make it more different from PageRank and HITS than most people probably realize. With more research, especially with input from a lawyer or law student who has some actual "human-intuition" of what makes a case more important than another, the algorithm presented here could become even more accurate.

I'm disappointed that my original idea, the idea that gave me the idea for this entire project in the first place, didn't work out. I'm not completely convinced that a ranking using something along the lines I described will *never* work, but there's definitely several problems that need to be addressed. If I had more than roughly a month to do this project, perhaps I could of had a breakthrough. Regardless the knowledge I gained from trying is priceless.

If anyone reading this (I always feel like teachers never actually read papers from students, which then makes all the work feel utterly worthless, which then leads to apathy and me not being motivated. This feeling is also why I feel safe putting this little run-on rant in at the end of the paper, just to see if anyone actually reads it) is interested or has questions about the legal project this paper used to perform tests on, contact Doug Niehaus.

8 Bibliography

[1] Brouwer, Andries. Haemers, Willem. Spectra of Graphs. 2008.

[2] Geng, Xiubo. Query Dependent Ranking Using K-Nearest Neighbor. ACM. 2008.

[3] Granqvist, Hans. More Like This, Building a Network of Similarity. http://techblog.netflix.com/2011/04/more-like-this-building-network-of.html April 18, 2011.

[4] Hardesty, Larry. Drug discovery, Netflix style? http://web.mit.edu/newsoffice/2010/drug-development-0413.html April 13, 2010.

[5] Jiang, Bian. Ranking with Query-Dependent Loss. ACM 2010.

[6] Langville, Amy. Meyer, Carl. Google's PageRank and Beyond: The Science of Search Engine Rankings. Princeston University Press. 2006.

[7] Spielman, Daniel. Spectral Graph Theory: Chapter 16 of Combinatorial Scientific Computing. Yale University.

[8] West's Encyclopedia of American Law. Court Opinion, Stare Decisis, Federal Court. Thomson Gale. 2010.

9 Appendix A: The Code

9.1 The Query-Independent Ranking

The following code is a python implementation for the algorithm specified at the end of section 4.2. It uses a built-in python module, *numpy*, to do efficient matrix operations. If, later on, a more useful starting vector, r_0, is to be used, another parameter would need to be added. Currently, the starting vector is statically set to e/n, essentially giving each case an equal ranking to begin with.

```
import numpy
numpy.set_printoptions(precision=15)    # Print to 15 decimal places
numpy.set_printoptions(suppress=True)   # Dont use scientific notation

def case_rank(Omega, iteration_cap=15, xi=0.95):
    """
    Given an n x n Omega (numpy) matrix, computes the rank for each case.
    Returns a tuple of three numpy arrays. The first is the final ranking
    vector, r^T. The second is the inlink ranking vector x^T. The third
    is the outlink ranking vector y^T.
    """
    Omega_T = Omega.transpose()
    n = Omega.shape[0]
    e = numpy.ones(n).reshape(n, 1)
    r = e / n
    xi_constant = ((1-xi)/n)*e

    for k in range(iteration_cap):
        x = xi*numpy.dot(Omega_T, r) + xi_constant
        y = xi*numpy.dot(Omega,   r) + xi_constant
        r = x + y
        r = r / sum(r)

    return r.transpose()[0], x.transpose()[0], y.transpose()[0]

"""
The Omega matrix for the example in section 4.2 is:

    >>> ex_Omega = numpy.array([[0.00, 0.00, 0.00, 0.00, 0.00],
                                [0.25, 0.00, 1.00, 0.00, 0.00],
                                [1.00, 0.00, 0.00, 0.00, 0.00],
                                [0.50, 0.25, 0.75, 0.00, 0.00],
                                [0.00, 0.50, 0.00, 0.00, 0.00]])

and the rankings can be found by:

    >>> r, x, y = case_rank(ex_Omega)
    >>> print r

To get the same values displayed in that section:

    >>> r, x, y = case_rank(ex_Omega, 2)
    >>> print r
    [ 0.2260896   0.21833964  0.29541638  0.19745337  0.06270101]
"""
```

9.2 Building an Omega Matrix

The following code is a python implementation for building an Omega matrix to use in my *case_rank* function. This code is not a general solution. It pulls data from a MySQL database using a schema specified by the *Legal Informatics Group at KU*. The overall approach, however, should be relatively the same for any database structure used.

```
import numpy

# This more or less sets up the database connection and
# imports the models/functions that will be used to query
# the Case table.
#
import djangobridge.lig as dbridge
dbridge.init_support()
from legalsources import models

# Initialize Omega to an n x n matrix of all 0's
#
n = models.Case.objects.all().count()
Omega = numpy.zeros(n*n).reshape(n,n)

# Loop over all the cases
#
cases = models.Case.objects.all()
for case in cases:
    # For each inlink, set the corresponding entry in the Omega matrix
    # to the weight of the link. If we only look at inlinks, eventually
    # all the outlinks will also be taken care of, since, at some point,
    # every outlink is an inlink to some other case.
    #
    for citing_case in case.citing_refs.cases():
        # set the citing_case's inlink omega value in the Omega matrix
        #
        Omega[citing_case.source.id-1][case.id-1] = citing_case.weight/4.0

# At this point the variable Omega contains the weighted adjacency
# matrix that can be passed in to the case_rank function.
print Omega

"""
Note: This code assumes a few things that aren't necessarily true outside
of my test cases.

First, it assumes that the database IDs of all cases are sequential and
there are no gaps. In other words, if there are N cases, it expects the
database IDs to be 1 to N and NOTHING ELSE.

Second, it assumes that there is always a weight available for the
citing case and that that weight is either 1, 2, 3, or 4. In other words,
no error checking is done to make sure the weight field is populated
much less legally populated.
"""
```