

# A Competitive Measure of Search Engine Relevance

Gordon Rios, Inktomi Corp.  
grios@inktomi.com

## Abstract

In this article I present a simple but powerful competitive model for evaluating search engine relevance. The model makes use of ranking and grading data in progressively more realistic models of user behavior. After building a markov matrix of the transition dynamics we are able to develop the long run equilibrium user share for each engine *independent* of the initial user distribution. The approach specifies a standard interface to the results of any query based relevance test. A key implication is that the market for web search may not be a *commodity* market as is often supposed. Instead, pockets of differential relevance across queries may provide enough *niche* possibilities to support a rich variety of web search providers.

## 1 Introduction

Consider a process of evaluating the relevance of documents returned by a set of search engines for a given set of queries. There are at least two fundamental analyses that can be performed:

- Independent: Each engine can be evaluated for average relevance across the set of test queries. For example, two engines can have similar numbers of *Excellent* documents but no attempt is made to analyze the overlap of *Excellent* documents across the two engines.
- Interdependent (or competitive): Each engine's results are evaluated with consideration given to *alternative* search engines for the same queries. For example, two engines  $E_1$  and  $E_2$  return the same number of *Excellent* documents but  $E_2$  returns them for twice as many queries.

The key benefit of using a competitive measure is that we can estimate the impact of changes to our engine or the competitive environment (new search engines) within the same consistent framework. Ultimately, this yields information that most effectively helps build and sustain competitive advantage.

## 2 Relevance Testing Framework

The basic tool for evaluating search engine efficacy is the *Relevance Testing Framework*<sup>TM</sup> (RTF). The RTF uses statistical samples of queries, evaluated across a set of search engines, to evaluate the absolute and relative effectiveness of the search engine. For each query a set of documents is assembled which is the union of the top  $K$  results from engine test set. The documents are evaluated by a panel of editors who grade the documents for relevance and then rank the documents the subset of documents that are judged to be relevant for the query. At the end of the process each set of results for a given query-engine pair have been annotated with a grade (or pair of grades for *double blind*) and a rank within all the documents returned for the query.

## 3 Model Based Measurement Framework

The general report for the RTF produces statistics concerning absolute numbers or percentages of *Excellent* grades for a given engine's results. In terms of the rank judgements we are mostly concerned with percentages of top  $k$  results accumulated by the engine - in effect, a document ranking in the top  $k$  is evaluated as being highly relevant. Of course, even higher quality evaluations can be made by taking the intersection between the set of top ranked documents and those with high grades. In any case, these measurements allow direct comparison between any given set of search engines.

Ultimately, we are concerned with the relevance of a given search engine *relative* to the set of alternatives. For this analysis it's necessary to consider relevance on a *per query* basis and explicitly price underperformance on queries for which better alternatives exist. Effectively, we are trying to create a measure that captures the interdependent nature of searching on the web; to this end, let's introduce the notion of a transition model of user behavior.

### 3.1 Transition Models of User Behavior

Let us suppose that for a given test, we have a pool of searchers randomly drawing from the set of test queries perhaps according to some probability distribution on the queries reflecting the true query frequency on the web. Further, assume that they evaluate the query results of that engine and either stay or make a successful transition to an alternative engine with better ranked results. If we aggregate over all the queries with this simple model then we can characterize this process with a single step Markov model parameterized with a *transition probability matrix*  $P$ . We then analytically develop the steady state or long run equilibrium of the system for an estimate of user share for each engine.

Of course the analysis is sensitive to the specific model we choose for user behavior. Our results will be specifically true only for users following this model. However, we can use patterns already observed to specify a simple class of models. For example, we find satisficing behavior in the user click data - that is, users tend to stop searching upon finding a *good enough* result. We can also use our judged and ranked results and assume that a user always prefers a lower ranked document. Even if this is not strictly true for closely ranked documents it's certainly a tenable assumption.

### 3.2 Transition Probability Function

Each entry  $P_{ij}$  in the transition probability matrix is the result of a function  $p(i, j)$  describing the probability of transition from engine  $i$  to engine  $j$ . We are free to specify this function in any way so long as the rows of  $P$  sum to unity. In specifying  $p(i, j)$  we are developing our model of user behavior - when applied to the population of users the probabilities can be viewed as proportions of the user base that *flow* between different states or engines.

### 3.3 Analytic Steady State

Using results of matrix theory and markov chains we can analyze  $P$  to develop the long run equilibrium distribution of probabilities or proportions of users across the engines. Since each step of the dynamics just post multiplies the current proportions by  $P$  we want the asymptotic value of a recursion that looks like  $\pi_{t+1} = \pi_t P$  where  $\pi_t$  is a  $1 \times m$  vector of user proportions at time  $t$  and  $P$  is the  $m \times m$  transition probability matrix. As  $t$  (assumed to be a discrete time step) goes to infinity we have  $\pi_0 P \cdot P \cdots P = \pi_0 P^\infty \rightarrow \Pi$  the long run equilibrium proportion of users at each engine. Under the conditions we have already described for  $P$  our  $\Pi$  is simply the eigenvector of  $P^T$  with associated eigenvalue of one. The existence of this eigenvalue is guranteed by theorem and  $\Pi$  is independent of the initial probabilities or user proportions.

## 4 An Application of the Framework

We now apply the framework to a specific case. First we specify a user behavior model that leads to a specific TPM for the problem. Then we solve for the long run equilibrium as described in the prequel.

### 4.1 A Simple User Transition Model

Our simplest model is that a user randomly draws from our pool of queries and evaluates the *best* or lowest ranked document in the top  $k$  results for that query for a given engine. Then, based on the absolute grade of the document, the user has a probability of evaluating other engines until he finds a lower ranked result. If the user finds a lower ranked result he makes a transition, otherwise he stays at the current engine.

### 4.2 A Specific Transition Probability Function

Taking the model outlined above we can derive a function for  $p_q(i, j)$  for query  $q \in Q$  engines  $i, j \in E$  where  $Q$  is the set of queries and  $E$  is the set of search engines. There are two cases to consider:

1. Search engine  $i$  does not have the best (or lowest ranked) document for the query

$$p_q(i, j)_{j \neq i} = p_{i,q}(S) \cdot \frac{I(R_{j,q} < R_{i,q})}{\sum_{i=1}^N I(R_{i,q} < R_{i,q})}$$

$$p_q(i, i) = 1 - p_{i,q}(S)$$

2. Search engine  $i$  has the lowest ranked document (e.g.  $R_{i,q} \leq R_{k,q} : \forall k \in E$ )

$$p_q(i, i) = 1 \text{ and } p_q(i, j)_{j \neq i} = 0.$$

Where  $R_{i,q}$  is the minimum judgement rank for the documents returned by engine  $i$  and query  $q$ ,  $p_{i,q}(S)$  is the probability that the user conducts the Search at an alternative engine, and  $I(c) \equiv \begin{cases} 1 & \text{when } c \text{ is true} \\ 0 & \text{when } c \text{ is false} \end{cases}$  is the indicator function. We then aggregate on the set of queries to compute  $p(i, j) = \sum_q p(q) \cdot p_q(i, j)$ .

In our example we simply look up  $p_{i,q}(S)$  from a table with probabilities assigned to different combinations of *grades*. For these results we used the following probability assignments.

| Paired Grades               | $p_{i,q}(S)$ |
|-----------------------------|--------------|
| <i>Excellent, Excellent</i> | 0.00         |
| <i>Excellent, Good</i>      | 0.10         |
| <i>Excellent, Fair</i>      | 0.25         |
| <i>Good, Good</i>           | 0.50         |
| <i>Good, Fair</i>           | 0.75         |
| <i>Fair, Fair</i>           | 0.90         |

### 4.3 Results On A Small Query Set

Running the framework on 94 queries chosen at random from various query logs we obtained the following distribution (see Fig. 1) for a small set of engines. Importantly, adding or dropping an engine does not significantly affect the relative performance of the remaining engines – only the absolute probabilities or proportions change. Of course, these results are only preliminary.