Abstract. It is a well-known statistical property that learning tends to slow down with each additional data point. Thus even if scale effects are important in web search, they could be important in a range that any viable entrant could easily achieve. In this paper we address these questions using browsing logs that give click-through-rates by query on two major search engines. An ideal experiment would be to fix the "query difficulty" and exogenously provide more or less historical data. We approximate the ideal experiment by finding queries that were not previously observed. Of these "new queries", some grow to be moderately popular, having 1000–2000 clicks in a calendar year. We examine ranking quality during the lifespan of the query and find statistically significant improvement on the order of 2–3% and learning faster at lower levels of data. We are careful to rule out alternate explanations for this pattern. In particular, we show that the effect is not explained by new, more relevant documents entering the landscape, rather it is mainly shifting the most relevant documents to the top of the ranking. We thus conclude they represent direct scale effects. Finally, we show that scale helps link new queries to existing queries with ample historical data by forming edges in the query document bipartite graph. This "indirect knowledge" is shown to be important for "deflating uniqueness" and improving ranking.

Scale Effects in Web Search

Di He[1], Aadharsh Kannan[1], Tie-Yan Liu[1], R. Preston McAfee[1], Tao Qin[1], and Justin M. Rao[2]

[1] Microsoft AI & Research
[2] HomeAway Inc

Keywords: scale effects, direct effects, indirect effects, intent clustering, unsupervised learning, web search

1 Introduction

A key question in the analysis of web search markets is the degree increased scale confers a direct performance imagine advantage. Consider two entirely different worlds. In the first, ranking quality is driven overwhelmingly by algorithmic innovation and fixed document features. In this world, a well-funded new entrant could potentially produce results of quality superior to the entrenched market leader. In the second, learning from historical queries is critical to ranking quality. A superior, but data-starved algorithm could perform much worse than the incumbent's. Although these two worlds are dramatically different in terms of the potential for innovation and competitive dynamics, little is known about which one we live in. Further, it is a well-known statistical property that learning tends to slow down with each additional data point. Thus even if scale effects are important, the steep part of the learning curve could be in a range that any viable entrant could easily achieve.

In this paper we address these questions using browsing logs that give click-through-rates (CTR), a natural measure of whether or not a set of results met the user's need, by query on two major search engines. We start by documenting the fact that more common queries indeed have higher CTR. The relationship is proportional to the square root of the log of historical clicks, indicating that increases are higher at lower data levels. Both search engines show similar functional forms.

These high-level correlations cannot be viewed as causal relationships because more popular queries could be innately easier to satisfy user intent. An ideal experiment would be to fix the "query difficulty" and exogenously provide more or less historical data. This is, of course, not possible. We approximate the ideal experiment by finding queries that were not previously observed. Of these "new queries", some grow to be moderately popular, having 1000–2000 clicks in a calendar year. We examine ranking quality during query lifetime and find statistically significant improvement on the order of 2–3%, with faster improvement at lower levels of data. We are careful to rule out alternate explanations for this pattern. In particular, we show that the effect is not explained by new, more relevant documents entering the landscape, rather it is mainly shifting the most relevant documents to the top of the ranking. We thus conclude they represent direct scale effects.

The fact that learning is fastest at low levels of data is a double-edged sword for a potential entrant. On the one hand, it seems to indicate that only a modest scale is required to achieve viability. While this is good news for relatively popular queries, which do account for a majority of *searches*, it is bad news for rarer queries, which account for the majority of *queries*. For example, in 2007 Google reported that 20–25% of the queries they see each day are unique when compared to the most recent month.[3] Moreover, most users submit at least some "long tail" queries [6].

The issue of long-tail queries adds a nuance to our analysis. If most queries really only have no more than five historical examples, then perhaps scale does not play much of a role after all. However it has been shown that historical examples of related queries can be linked to seemingly rare queries by applying clusters and graph cutting techniques to the query-document bipartite graph [2][9][8]. This graph can be used to generate related queries and leverage historical examples that differ in minor ways from the target query. To understand the role of scale in this domain, we apply a clustering algorithm motivated by past work. To do so, we take the query-document graph—the total nodes number nearly 10 billion—and cluster queries that share the same intent. Human evaluation is used to validate the accuracy of the algorithm.

[3] http://searchengineland.com/that-25-new-queries-figure-ballpark-estimate-says-google-11596

We use the graph to flexibly "deflate uniqueness" because it creates ties between relatively rare queries to more common queries that capture similar user intent. We show, for instance that for a set of 1.1 billion "long tail" queries, there are 10-fold less unique instances of intent. The method naturally surfaces synonyms and related queries. Experiments reveal that increasing overall scale provides greater edge density, which in turn allows one to link more rare queries to more common queries with many historical examples. In summary, this analysis shows that there are additional returns to scale in the form of semantically linking queries and that queries submitted by users are "not as unique as they appear."

Finally, we conclude with some thoughts on the larger picture. Our analysis here is not capable of capturing all the returns to scale, rather we focus on clean identification in relatively controlled environments. That being said, it is important to note the CTR impact of scale we document appears modest overall, order 2–3% of CTR. Interpreting magnitudes, however, is a bit tricky. For example, both providers have CTRs on tail queries in the 70% range. Suppose an entrant could achieve 60% "off the shelf." Then 2-3% represents more like 20-30% of the meaningful range in which we expect competitors to be differentiated and thus appears quite large in this light. We stress that this is only an example to highlight the nuances in interpreting the magnitudes reported in our study.

2 Data Description

Our data consists of search logs for a period of time greater than 6 months from two large commercial search engines. The source are proprietary logs of a web browser. In all instances, the same restrictions are applied to both search providers. For example, the same user types, geographic locations, and so forth. Table 1 shows that we observe hundreds of billions of searches. This richness will allow us to conduct a detailed analysis of queries as data accumulates over time.

Table 1. Summary statistics

Provider 1 (# impressions)	>200 billion
Provider 2 (# impressions)	>300 billion
Provider 1 # clicks	>100 billion
Provider 2 # clicks	>150 billion

3 Direct Effects of Scale

In this section we study how the search engine performance is related to the volume of historical data for a target query. We first investigate all the queries in our dataset and then check those relatively new queries, which have only a few historical clicks.

3.1 Analysis of General Queries

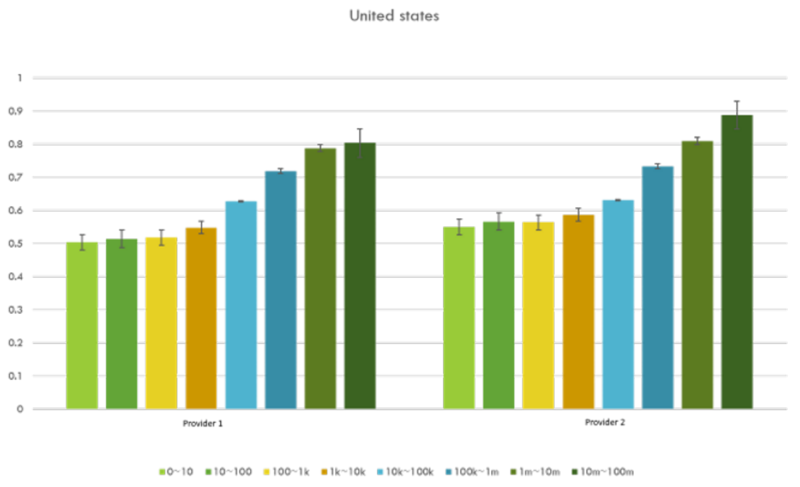
To study the scale effect of a query, ideally, we need to collect its search log from the first time it is observed, and see how its CTR changes as more people issue it. However, this is infeasible because that a search engine can only legally keep the data for a limited time[4]. Thus for popular queries, it is hard (if not impossible) to know its first appearance. Here we use one year as the range of time for analysis, and collect browsing logs of two commercial search engines, i.e., Provider 1 and Provider 2. For the log in each provider, we use the first three month's data as the *benchmark* data source, which acts as the data observed in history, and use the remaining nine months of data as our *target* data source in analysis. For each query q and each day d in target data source, we get a pair $< H(q,d), CTR(q,d) >$ in which $H(q,d)$ denotes the historical measure

[4] http://searchengineland.com/google-responds-to-eu-cutting-raw-log-retention-time-reconsidering-cookie-expiration-11443

before day d for query q, and $CTR(q,d)$ denotes its CTR in day d. In the experiment, we use click number as the historical measure, since clicks are the most effective feedback from search users.

For each query, we generate 270 pairs and partition the pairs into buckets according to $H(q,d)$. The CTR averaged over the pairs in each bucket is shown in Figure 1. We can see from the figure, CTR shows a positive correlation with the number of historical clicks for both the search providers, and the patterns of CTR growth of the two providers are similar.

Fig. 1. In aggregate, CTR shows a positive correlation with the number of historical occurrences. Both providers show a similar relationship.



To quantitatively characterize the scale relationship, we further conduct a regression analysis on the correlation between the CTR and the number of historical clicks. After trying several different function families, including linear functions and polynomial function, we find that the square root of the log historical clicks well approximate the current CTR. Specifically, denote the historical click number as x, we have that for provider 1

$$CTR = -0.0530[-0.085, -0.021] + 0.3287[0.315, 0.343]\sqrt{log(x)}, \tag{1}$$

and for Provider 2,

$$CTR = -0.3871[-0.486, -0.288] + 0.4792[0.438, 0.520]\sqrt{log(x)}. \tag{2}$$

This shows that there is a strongly positive dependency between the number of historical clicks and the current search performance. These results can be seen graphically in Figure 2 and Figure 3.

3.2 Scale Effect Analysis on New Queries

One a major concern with the analysis in previous subsection is that the queries falling into different buckets are not the same. For example, popular queries may express intent that is innately easier to satisfy. Since these more popular queries would fall into right-side buckets and rare queries into left-side buckets, the correlation we observe could be due to this confound and not a direct impact of scale. An ideal experiment would be to fix the "query difficulty" and exogenously provide more or less historical data. Since this is not possible, we approximate the ideal experiment as follows.

We select a set of queries according to two criteria: (1) a query has less than 200 clicks in the three-month benchmark data source; (2) the total number of clicks of the query in the calendar year (including both the

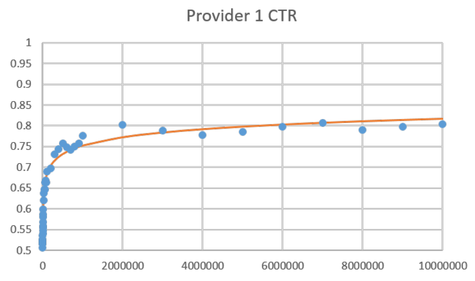


Fig. 2. Provider 1, relationship between CTR and number of historical examples.

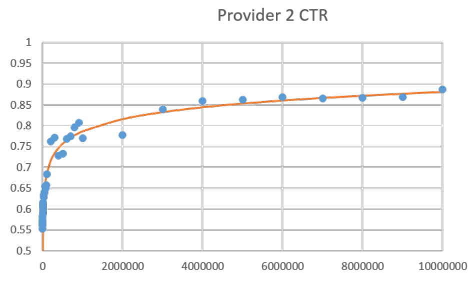


Fig. 3. Provider 2, relationship between CTR and number of historical examples.

benchmark data source and the target data source) is between 1000 and 2000. The first criterion ensures that such a query is relatively new to the search provider, and the second criterion tries to make that the selected queries have the similar difficulty the search provider. As a result, there are about 8000 queries selected for Provider 1 and 10000 for Provider 2. Because we see almost all the queries for one provider, and a much smaller fraction for the other, the scales are not directly comparable.

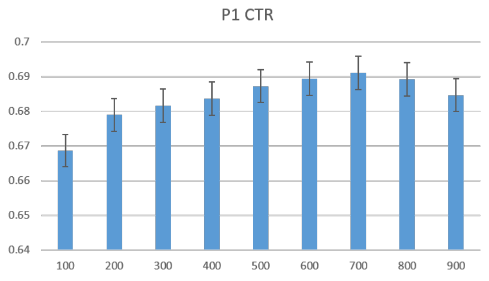


Fig. 4. Provider 1, relationship between CTR and number of historical examples for new queries only

For query q, we use $CTR(q, c)$ to denote the CTR of q in period of receiving $c+1$ to $c+100$ clicks. For each selected query q, we get 9 pairs $< c, CTR(q, c) >$, where $c \in \{100, 200, \ldots, 900\}$. Then we partition the pairs into buckets according to c and calculate the average CTR over queries in each bucket. It is important to note that the queries in each bucket are the same, meaning selection effects cannot drive observed relationship.[5]

[5] We do not include the pair $< 1000, CTR(q, 1000) >$ since not all the selected queries have 1100 clicks in the target data. If we include this pair, the queries in the last bucket will be less than the queries in the left 9 buckets.

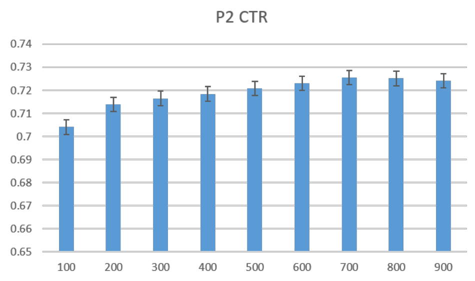


Fig. 5. Provider 2, relationship between CTR and number of historical examples for new queries only.

We present our aggregated results with error bar (confidence interval = 0.95%) in Figures 4 and 5. From the curves, overall we observe that CTR grows for new queries for both providers, and the growing trends are significant. This shows that the scale effect does exist in both search providers on the order of 2% over the first 1,000 queries. A regression of the same yields for Provider 1 an intercept of 0.6726 [0.6653,0.6787] and coefficient of 2.116 e-05 [1.03154 e-05, 3.2017 e-05] i.e. anywhere from 1-3% CTR gain per 1000. For Provider 2 an intercept of 0.7075833 [0.70145,0.99465] and coefficient of 2.083e-05 [9.94658 e-06, 3.172008 e-05] i.e anywhere from 0.99-3% CTR gain per 1000. This is the same order difference observed at the left-hand side of Figure 1, but it cannot explain the order 20% increase documented in the overall relationship. While we cannot rule out these are driven by scale effects, the evidence seems indicate that the large Figure 1 differences are more of a selection issue on query difficulty, as learning appears to slow down after 1,000 historical instances.

3.3 Robustness Checks

It is important to consider alternative explanations to direct scale effects. A natural alternative hypothesis is that improved performance is due to richer or more relevant documents, not better ranking. To see if this is going on, we revisit our new query analysis and tag URLs that were previously in crawled as "old." New URLs would be clicks on links that were not previously available to the ranker. If the growth in CTR was due to new URLs, we should see the fraction of old URL clicks decrease as we move to the right in Figure 6. Instead we observe that a constant and very high, about 98%, of clicks are on old URLs. Since this fraction does not change, it is not able to explain the growth in CTR. Further we can look at whether the ranker is doing a better job at putting the best links at the top of the page. It is well-known in search that position causally influences CTR in a multiplicative fashion—placing higher quality links at the top of the page leads to a increase in user satisfaction [5]. We can only observe click position for one of the two providers that we have considered, but document a strong, statistically significant improvement with historical examples for a new query analysis.

A final robustness check is to consider the underlying causal model for why scale can directly improve results. First, papers have shown how features can be improved, such as including past queries as anchor text for clicked links [10]. That is, position changes as a result of user behavior. These data can then feed into the creation of "click graphs" [5, 7, 8] which is useful for building out semantic knowledge around a query and user level analysis to understand intent satisfaction [3]. Thus far from being a black box, there are many previously identified causal channels that use historical behavior to directly improve ranking performance.

4 Indirect Effects of Scale

In this section, we explore the effects of data on related queries, which supplement direct query data. To do so, we first identify related queries by constructing a knowledge graph. We use the cosine measure of relatedness, and then classify as related or not by the measure exceeds a threshold. To set the threshold, we use human judgment on a subsample. We then explore how related query knowledge affects the CTR using regression techniques.

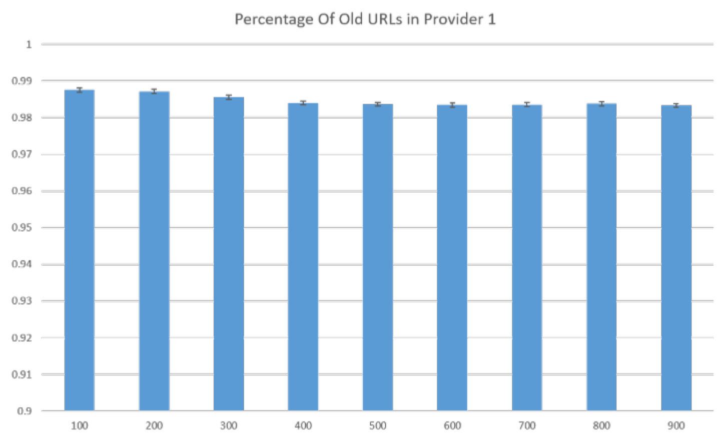


Fig. 6. Provider 1, the fraction of clicks that correspond to URLs previously observed. Results indicate a stable and very high percentage of clicks comes from these documents.

Our data for this section consists of search logs for a period of time greater than 6 months from a large commercial search engine. The data form a bipartite graph of queries, denoted $\mathcal{Q}$, and documents, denoted $\mathcal{D}$. Edges are given by the set $\mathcal{E}$ represent user clicks connecting queries to documents. Edge weights are given by the click count $C_{i,j}$ from query node i to document node j. We combine queries that only have slight differences in lexical form. To do so we follow standard best practices for normalizing queries: 1) Eliminate any punctuation marks 2) Split queries into the words (which include numbers) 3) Porter stem words (remove plurals and other standard stemming operations) to eliminate differences in cases 4) Represent each query as a bag of the remaining words, sorted alphabetically. This procedure reduces the number of query nodes by 7%.

Table 2 summarizes the graph and the underlying user activity. The graph has over 7 billion nodes connected by over 11 billion edges. The total number of clicks exceeds 100 billion. These statistics highlight the scale of modern search engines and also point to the sparsity of the graph.

Table 2. Summary statistics for the query-document graph and underlying user activity

Cardinality $\mathcal{Q}$ (# unique queries)	4.82 Billion
Cardinality $\mathcal{D}$ (# unique URLs)	3.26 Billion
Cardinality $\mathcal{E}$ (# edges)	11.6 Billion
Number of sessions	$>$ 100 billion
Total clicks	$>$ 100 billion

4.1 Core Algorithm

We start with the query-document bipartite graph. This can be represented by a matrix with dimensions $card(\mathcal{Q}) \times card(\mathcal{D})$. Each column gives a vector for each document where the jth entry gives the clicks from query j. In other words, it gives the document's representation in query space. For every pair of documents, we compute the cosine distance to form an upper triangular document similarity matrix. This requires order $\frac{card(\mathcal{D})^2}{2}$ calculations. Next, we convert similarity weights to 0 or 1 using a chosen threshold; this censoring removes weak ties and allows us to form a document similarity graph (we implement multiple thresholds and use human accuracy ratings and other metrics to find the preferred setting). We find the connected components from this graph [1] and call them *intent clusters*. Intent clusters capture groups of documents that have the same inferred intent because users clicked from similar queries to get to these documents.

Finally, we take intent clusters and form the query-intent-cluster bipartite graph. Edge weights are given by the fraction of clicks from query q that are point to a document in cluster c. Edge weights have the natural interpretation of the fraction of searches for a given query that had a given intent. If 10% of clicks from query X map to g_1 and 90% map to g_2, then we say that query X has intent g_2 and g_1, with weights 90% and 10% respectively. We will observe that this is very common. It is natural to label each intent cluster as the query with the highest weight, which we call the "intent query."

Computing cosine distance is straightforward, but given the query-document bipartite graph has dimensions $card(\mathcal{Q}) \times card(\mathcal{D})$, doing so requires order $\frac{card(\mathcal{D})^2}{2}$ calculations. Since we implement our approach on a modest-sized compute cluster, the parallelizable nature of this computation makes it feasible even though we have billions of nodes.

Using these distance calculations, we form upper-triangular document similarity matrix. The next issue we have to address concerns the fact that clicks are a form of implicit feedback that contain noise—some clicks do not represent user intent. This means entries in the similarity matrix are biased away from zero as compared to ground truth. This points to the use of a threshold wherein similarity scores below the threshold are given the value 0 and those above are given the value 1. Once values are converted to a binary indicator, the matrix is converted into document similarity graph for which we can conduct a connected components analysis using a scalable algorithm.

Ex-ante it is not obvious what value of the threshold is optimal. Thresholds that are too low induce noise and could lead to massive connected components that do not represent one true intent. Thresholds that are too high could lead to too sparse a graph, meaning many clusters actually have the same underlying intent. Based on pre-testing, we choose 4 threshold values: 0.70, 0.80, 0.90 and 0.95. We will later show that human judgment can help select the optimal threshold

In order to compute connected components we follow a simple strategy of iterative agglomeration. One can conceptualize this strategy as (a parallel) flood fill algorithm on a Map-Reduce framework. For the first iteration we treat every document pair (from the similarity matrix) as a separate cluster, identify link nodes between pairs and merge them. Convergence of link node identification and merging clusters for subsequent iterations is reached through repetition.

Referring to Algorithm 1. SetOfURLPairs contains a LeftURLId, a RightURLId and RowLabel. LeftURLId and RightURLId are both ordinal numbers identifying an individual URL. Each one of these elements represents a non-zero entry in the URL similarity matrix where LeftURLId < RightURLId. RowLabel is an ordinal number corresponding to every entry in SetOfURLPairs. RenameIfExist replaces the ClusterId with the SlaveCluster value if there exists an entry in SetOfClusterRenames where MasterCluster = ClusterId else it returns ClusterId. MaxIterationLimit is a computation limit that is set to avoid infinite non-convergence. The query node identification and merging is the process that is repeated till convergence. We found that the algorithm typically converged in 5–6 iterations.

Let's call the set of connected components $\mathcal{G}$ with elements g, which itself is a set containing the documents within each component. We now form the query/intent-cluster graph. For each query q, the edge weight to intent cluster g_i is the fraction of clicks from that query that point to node g_i. This weight has a natural Markovian interpretation. For purposes of semantic interpret-ability, we label each node with the query that has the highest edge weight pointing to that node. We call this the *intent query*. Traversing the graph following the Markov weights reveals related intent, an approach that has been shown to be useful on the raw query-document graph to find related queries [4].

4.2 Evaluation of clusters

Our goal in this section is to get a sense to what extent do the *intent queries* reasonably capture the intent of the user. To do so, for each threshold setting we form a 100-query test set by randomly selecting 10 from each decile of the search frequency (the same queries are used for each setting). We then follow all the edges in the knowledge graph to get all the intent queries (clusters) that map to this query. Note that we follow all edges, even if the Markov weight is very low.

An independent auditor, blinded to the parameter settings or aims of the study, evaluated query/intent-query pairs. Pairs were scored a 1 if the intent query would could reasonably match the underlying query. The auditor used the appropriate references for queries she was not familiar with. For example, for the query "Aretha Franklin" there is a link to "Luther Vandross" intent cluster. The auditor scored this as a

ALGORITHM 1: Find connected components

```
SetOfURLWithCluster =
SELECT LeftUrlId AS UrlId,
        RowLabel AS ClusterId
FROM SetOfURLPairs
UNION
SELECT RightUrlId AS UrlId,
        RowLabel AS ClusterId
FROM SetOfURLPairs;

for 1 TO MaxIterationLimit do
  SELECT UrlId AS LinkUrl,
          MIN(ClusterId) AS MasterCluster
  FROM SetOfURLWithCluster
  FOR EVERY LinkUrl;

  SlaveClusters =
  SELECT ClusterId AS SlaveCluster,
          MIN(UrlId) AS LinkUrl
  FROM SetOfURLWithCluster
  FOR EVERY SlaveCluster;

  SetOfClusterRenames =
  SELECT MasterCluster, SlaveCluster
  FROM MasterClusters
  INNER JOIN SlaveClusters
  ON LinkUrl
  WHERE MasterCluster <> SlaveCluster;

  if SetOfClusterRenames.size = 0 then
    end for
  end if

  SetOfURLWithCluster =
  SELECT UrlId RenameIfExist(ClusterId, SetOfClusterRenames) AS ClusterId
  FROM SetOfURLWithCluster;
end for
```

0, concluding that while the two entities are certainly related—they are both singers of a similar style from a similar era—that the query "Aretha Franklin" does not reasonably have the intent to find material on Luther Vandross. Clearly there is some genuine ambiguity at play in how to make this judgment. On the one hand, some users may want to find material on Luther Vandross but are unable to remember his name. They search a name they can remember, namely Aretha Franklin, and then click on a document in the Vandross cluster. In this case, one might conclude that the judgment should be a 1. (In either case, these are clearly terms that are usefully "related to" the underlying query.) To push back against this issue, we simply asked our auditor to be conservative and consistent. Precision is defined as the fraction of pairs that are judged to be relevant to each other by this standard. Weighted precision applies a weight to each pair as given by the Markov weight connected the query to the intent cluster. This means that connections that had low strength are down-weighted in the calculation.

The parameter setting 0.7 produced the most edges in the knowledge graph, which is natural since it required the lowest threshold to establish similarity and thus the most non-singleton connected components. We define each query/intent-query pair for the 0.70 that is scored as a true positive as the target set. The fraction of pairs that each method recovers is defined as pseudo recall. By definition it is equal to 1.0 for the 0.7 parameter setting. We also define weighted recall, which applies the Markov weight to each pair. Note that now the measure is not constrained to [0,1]. The reason is that the tighter thresholds tend to lead to higher Markov weights, so if they can actually "recover more" (by getting credit for the weight) pairs. We concede this metric is a bit unconventional, but find it nonetheless informative.

Table 3. Precision and Pseudo Recall by Threshold

Threshold	Raw Precision	Weighted Precision	Pseudo Recall	W. Pseudo Recall
0.7	0.69	0.79	1	1
0.8	0.70	0.84	0.76	1.054
0.9	0.68	0.83	0.45	1.04
0.95	0.66	0.83	0.26	1.03

Table 3 gives the results. Raw precision is highest for the 0.8 threshold, coming in at 0.70, and actually lowest for 0.95 (but the overall distances are not large). This indicates that the clusters in 0.95 can be too specific and thus often don't capture the broader intent of queries linked to them. Weighted precision is again highest for the 0.8 setting, coming in at a healthy 0.84. As expected, raw pseudo recall falls as the threshold tightens, but the weighted metric is far more stable. Again, threshold 0.8 scores the best on this metric.

Overall the metrics indicate that our unsupervised algorithm achieves results that are deemed quite sensible when exposed to direct human judgment. The fact that raw accuracy was relatively stable, indicates that optimal choice of parameter will probably depend on other features of the output, which we'll now investigate.

4.3 Linking queries to leverage scale

In Figure 7 we plot the cumulative distribution of the count of intent clusters per query. For this figure and all the rest we plot all 4 threshold values. The first feature that is immediately apparent is that most queries map to a single intent cluster. This is especially true for the 0.95 setting, which has the sparsest knowledge graph. For all parameter settings, 90% of queries map to 2 or less intent clusters. That being said, the distribution exhibits heavy tails. We have censored the x-axis at 10, but it extends well into the 100's, which can be seen in Figure 8. The log-log density plot shows the familiar linear patterns of a heavy tailed distribution.

The number of intent clusters that a query maps to captures the diversity of intent for a given unit of expression. What the data reveal is that while most queries seem to have a single intent (this also due to the sparseness of the query-document graph, as previously mentioned), a non-negligible fraction have quite diverse intent. Indeed, subsequent analysis reveals these more diverse intent queries tend to have higher volume, in part because they are more generic ("Aretha Franklin" vs. "Aretha Franklin's second album").

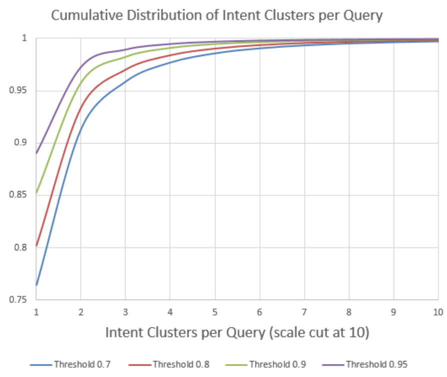


Fig. 7. CDF of of the number intent clusters with an edge to a submitted query.

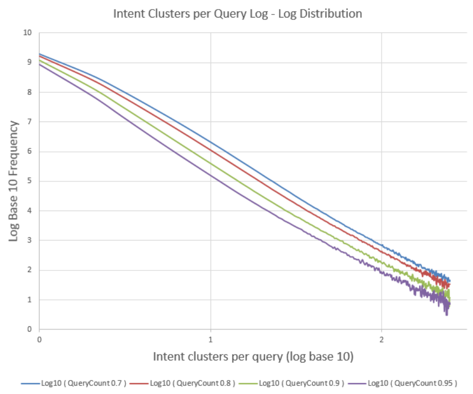


Fig. 8. PDF of of the number intent clusters with an edge to a submitted query, log-log scale.

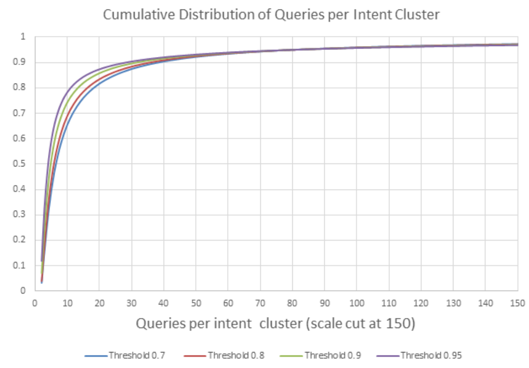


Fig. 9. CDF of the number of queries per queries per intent cluster.

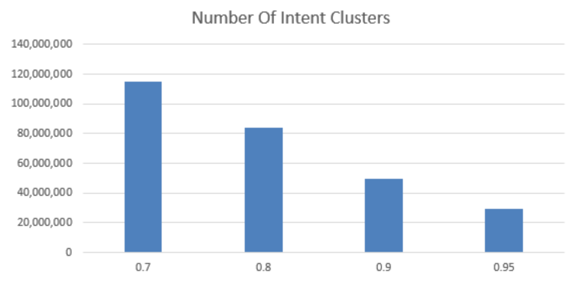


Fig. 10. Number of intent clusters per method.

One might wonder if the fact that most queries map to one intent query is a function of the fact that intent clusters tend to contain very few documents and thus very few queries linking to them. In Figure 9 captures how many in-links intent clusters have. The CDF reveals that yes, there are many small clusters. As expected, the 0.95 setting has the smallest clusters. However, for looser thresholds, most clusters have more than 5 queries linking in, and approximately 20% have more than 20 underlying queries. This shows that we have substantially reduced the sparsity of the query-document graph shown in Table 1. By reducing this sparsity, learning via historical examples becomes a much more promising avenue to improve search engine performance.

We have seen so far that the tighter the threshold, the fewer queries per cluster and fewer clusters per query. In Figure 10 we show how the number of non-singleton set clusters formed changes with each parameter setting. Examining the 0.7 setting, we see that there are roughly 120 million clusters. To put this number in perspective, we saw that roughly 75% of all intent clusters mapped to a single query, meaning they were a singleton set and excluded. The remaining set of queries, however, is quite large, 1.1 billion to be precise. Of these 1.1 billion "unique" queries, we identify that the underlying unique intent is 10-fold smaller. This is a substantial "reduction in uniqueness." The other thresholds deflate uniqueness more aggressively, but also leave more singleton sets, as shown in Figure 7. Given that human judged accuracy as similar across thresholds, these results point to using 0.7 as the threshold.

To summarize the results, we see that the choice of threshold has a large impact on the resulting knowledge graph. A smaller threshold allows more non-singleton intent clusters (connected components) to form in the document similarity graph. At first this might seem counter-intuitive. Since clusters are identified by connected components, adding more links could connect more existing components and reduce the number of clusters. However, the countervailing force is that in a graph this sparse and that displays so much isolation, adding links tends to form more clusters than it ties together. This highlights the role of scale in forming a richer graph.

The 0.70 threshold setting is shown to identify approximately 118 million intent clusters from the nearly 1.1 billion "unique" queries that link to more than one document. Indeed these are the queries that have the higher volume in terms of searches (almost by construction), so the "reduction of uniqueness" our method offers in terms of *search volume* is greater still. Figure 11 shows the relationship between direct and indirect volume (in log-log scale). We note that at low direct volume levels, indirect views are often two orders of magnitude greater, highlighting the importance of these links. We conducted experiments that artificially limited the scale of data we gave ourselves access to and saw the expected dramatic reductions in links within the query-document graph.

4.4 Impact on CTR

We now link the number of "indirect examples" to CTR performance by repeating the "new query analysis" using both direct historical examples and indirect historical examples (the queries linked in the graph) as features. Our findings are summarized as follows:

y_{ctr} – Success-CTR $\quad v_{id}$ – Indirect View Count $\quad v_d$ – Direct View Count

$$y_{ctr} = \alpha + \beta_1 v_{id} + \beta_2 v_d$$

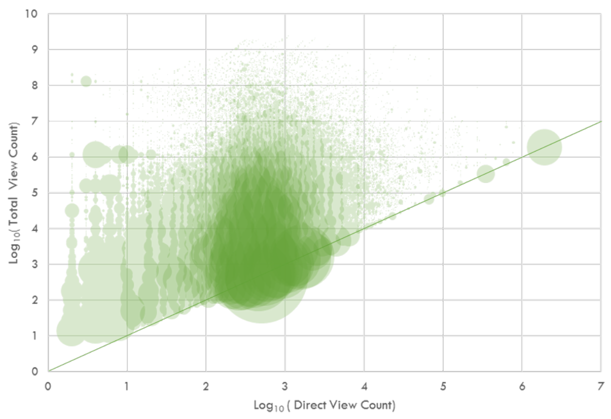


Fig. 11. Relationship between direct vs. indirect query counts

$$\alpha = 0.742 \; [0.740, 0.745]$$
$$\beta_1 = 2.251 \; 10^{-05} \; [2.79 \; 10^{-07}, 4.48 \; 10^{-05}] \text{ i.e. } 0.02\% \text{ to } 4.48 \; \% \text{ per } 1000$$
$$\beta_2 = 1.109 \; 10^{-05} \; [0.528 \; 10^{-06}, 1.69 \; 10^{-05}] \text{ i.e. } 0.5\% \text{ to } 1.7\% \text{ per } 1000$$

We find both (direct and indirect views) are statistically significant predictors of CTR and higher click positions on the page. For CTR, the coefficient on indirect views, conditional on direct views, is 0.000022, indicating that 1,000 indirect views predicts a rise in CTR of 0.02, which is consistent with our previous findings. Our previous finding was 1 to 3% and about 2%. This indicates that while direct examples are more important, leveraging related queries is an important factor as well. Given that scale increases the density of the query-document graph and thus the ability to find related queries, this points to another source of advantage conferred to scale.

5 Discussion and Conclusion

It is well known that, like most statistical learning problems, efficiently designed search engines have errors proportional to $n^{-1/2}$, where n is the amount of data. As data accumulates, search engines should improve, but how much does this scale economy matter? One perspective on competition among search engines is that even a 1% share represents billions of searches per year. But the scale of the problem solved by modern search engines has grown along with the data. Where AltaVista indexed millions of search pages, modern search engines index billions of pages. So while modern search engines have more data than they did a decade ago, they solve a harder problem than they did, making it entirely unclear whether the increase in scale makes the problem easier or harder.

The frequency of unique queries does not actually measure how hard the problem is, unfortunately. Consider the problem of "Pasadena Ethiopian Restaurant." The first time this query was entered (as far as we can tell), both search engines provided excellent results. Why? Because there are nearby queries; essentially all portions – Ethiopian, Pasadena and Restaurant – are understandable based on past data, and search engines can identify the relevant documents even though the query is rare. This spillover of knowledge from one query to another means that the counts of queries is a flawed measure of the data advantage. We apply an approach motived by past work that directly document these effects.

To address the complexity of the problem solved by search engines, we use two strategies. First, we look at new queries. These are queries that have been rare and then become much more common. This lets us see how the search engines respond to new data. We find that both search engines improve significantly as more data flows in. While we illustrate that process quantitatively, there are two caveats. First, we see almost all the data for one of the search engines and much less for the other. Thus, the scale of the measurement varies across the two engines. Second, it could well be that there are other queries bringing relevant data to the problem, because they help the search engine improve the matching not just for one query but for a set of queries. Nevertheless, we do find substantial improvement on rare queries as more data flows in, which demonstrates that both search engines are data starved in the sense that they benefit significantly from more data on a substantial portion of the queries, perhaps as many as half the queries and 15% of the searches.

One potential objection to the approach we take is that perhaps the available results get better as more people enter a specific query, because web pages specifically constructed to be clickable are created. We show that this is unlikely—around 98% of the pages already existed—but more work could be done on this topic. Indeed, the ecosystem effects of consumer search and gaming of search engines remain interesting topics beyond the scope of this paper.

To address knowledge spillovers, where data from "Pasadena restaurant" helps a search engine with "Pasadena Ethiopian Restaurant," we constructed a knowledge graph. The knowledge graph identifies related queries and lets us identify both direct (searches for that query) and indirect (searches for closely related queries) data that can be brought to bear in finding the right answers to a query. One finding that suggests that the knowledge graph was well-constructed is that we can predict the click through rate as a function of both direct and indirect data, and find both are relevant, with similar coefficients.

The knowledge graph model confirms that data—both direct and indirect— matters at scale. Moreover, and more interestingly, it lets us quantify how many queries have a modest amount of total data. We find approximately 10% of the queries have less than 1000 relevant observations, and 18% have less than 10,000. This addresses the question of how much data can actually be brought to bear on answering rare queries.

Search engines are arguably one of the most complicated engineering tasks ever attempted, matching billions of queries to billions of web pages. While there are probably increasing returns to scale for small amounts of data, there are eventually diminishing returns. Many markets are characterized by two major search engines, with one larger than the other. What we observe in the North American market is that both search engines are well into the region of diminishing returns, but there is still a significant return to data. The effects we estimate are of modest size, 1-4 percentage points, meaning that a major algorithmic improvement could swamp the advantage of a larger incumbent. That being said, it is well known that effects of this size are large in terms of differentiating performance from a competitor and thus strongly suggest that major search engines still operate in a region where more data matters.

Bibliography

[1] Baeza-Yates, R., Tiberi, A.: Extracting semantic relations from query logs. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 76–85. ACM (2007)

[2] Beeferman, D., Berger, A.: Agglomerative clustering of a search engine query log. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 407–416. ACM (2000)

[3] Bordino, I., Castillo, C., Donato, D., Gionis, A.: Query similarity by projecting the query-flow graph. In: Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval. pp. 515–522. ACM (2010)

[4] Craswell, N., Szummer, M.: Random walks on the click graph. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 239–246. ACM (2007)

[5] Craswell, N., Zoeter, O., Taylor, M., Ramsey, B.: An experimental comparison of click position-bias models. In: Proceedings of the 2008 International Conference on Web Search and Data Mining. pp. 87–94. ACM (2008)

[6] Goel, S., Broder, A., Gabrilovich, E., Pang, B.: Anatomy of the long tail: ordinary people with extraordinary tastes. In: Proceedings of the third ACM international conference on Web search and data mining. pp. 201–210. ACM (2010)

[7] Li, X., Wang, Y.Y., Acero, A.: Learning query intent from regularized click graphs. In: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. pp. 339–346. ACM (2008)

[8] Liu, X., Song, Y., Liu, S., Wang, H.: Automatic taxonomy construction from keywords. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1433–1441. ACM (2012)

[9] Sadikov, E., Madhavan, J., Wang, L., Halevy, A.: Clustering query refinements by user intent. In: Proceedings of the 19th international conference on World wide web. pp. 841–850. ACM (2010)

[10] Wen, J.R., Nie, J.Y., Zhang, H.J.: Query clustering using user logs. ACM Transactions on Information Systems 20(1), 59–81 (2002)