To build a robust search engine, efficiency is key. Inverted index is used to store the search query. The list of pages includes diverse results.

Search: apple

- Saccard $SS(apple, \{\text{words on page}\})$
  - cosine similarity

Spam pages: [apple, apple, ...]
  - hidden cause

Spam pages: copy highly ranked pages

apple > baby > child
apple pie
lyxi
Crawlers: programs that surf the web.

- Goes to page
- Reads; programs puts in index
- Finds links; follows links

\[ \text{\textless a href=} " \ldots " \text{\textgreater text a} \]

Spam pages: false pages point to forged pages.

Index Pages: curated list of links.
- Yahoo! / LookSmart

Page Rank

IDEA 1. Important pages are linked to by other important pages.

IDEA 2. Random Surfer: important if a random surfer goes to a lot

\[ G : (V,E) \]
- \( V \): pages
- \( E \): links between pages
\[ M = (V, P, g) \rightarrow g^x \]

Page \( n \in V \)

\( g^x(n) \rightarrow \text{rank value} \)

\( \text{Score } (u, "search") = \sum_{i \in u} (g^x(v), \cos (\text{link } u, v), \ldots) \)

Compute \( g^x \):

- MCMC, use crawler counts
- slow convergence
- \( P^x = P^x \cdot P^x \)
- \( P^x \) is super dense

- \( q^x_i = \frac{1}{\|P^x \|} \cdot \text{repeat } n \text{ times.} \)

- eigenvector
Teleportation
15% time -> jump to random page

$q_{i+1} = \left( (1-\beta)P + \beta Q \right) q_i$

\[ Q = \begin{bmatrix} Q_1 & \cdots & Q_n \end{bmatrix} \]

\[ q_i \sim 0.15 \]

Ergodic

\underline{Trust Rank}

Teleportation -> only jump to trusted pages

\[ P(v) \Rightarrow T(v) \]

"truth"

\underline{Personalized Page Rank}

id visits some pages regularly

\[ \Rightarrow \text{teleport to these} \]