Advances in Information Retreival

Dan Bănică
Eduard Băzăvan
Ciprian Pătru
Madi Petrăreanu
Alexandra Podiuc

Contents

- Understanding user behavior (eye tracking)
- User activity tracking
- Optimizing Search Engines using Clickthrough Data
- Using Lexical Chains for Text Summarization
- Multidocument Summarization – Centroid-based algorithm
- Graph-based Lexical Centrality
- Sentence Ordering

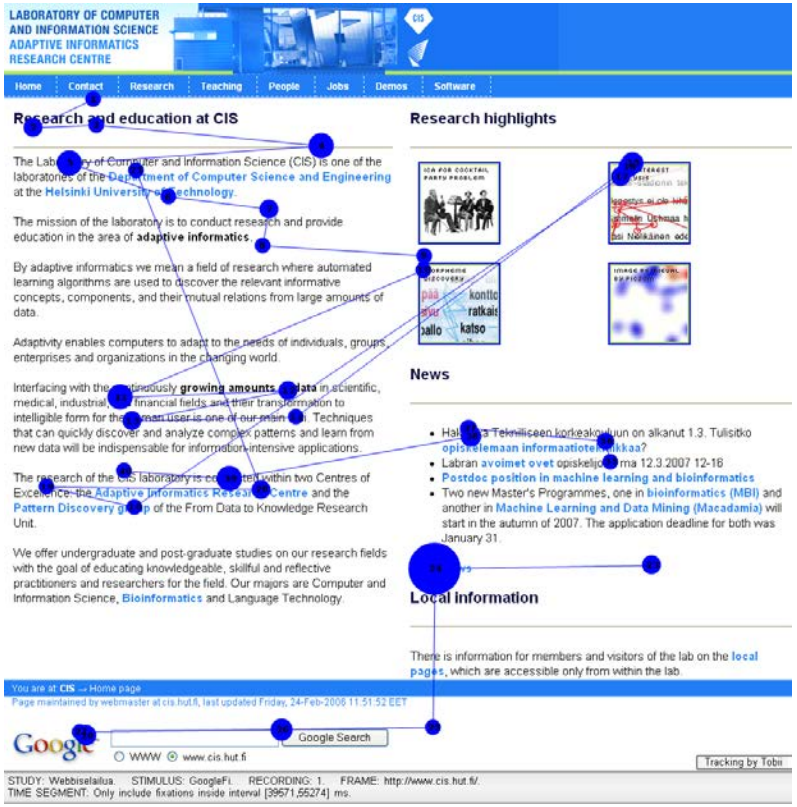
Understanding user behavior

Understanding user behavior

- Understand how to best satisfy the user
- Multiple studies based on eye tracking
- Users are very selective
  - How to better design a webpage?
  - How to model the attention of the user?
    - Ideally, an IR system should take this into account
  - How important is the ranking of a search engine?

Eye Tracking

- Fixations
  - The attention is directed towards a specific area of focus
  - 200-300 milliseconds (ms)
- Saccades
  - Rapid moves between fixations
  - 40-50 ms
  - Information processing is suppressed



Investigating online search

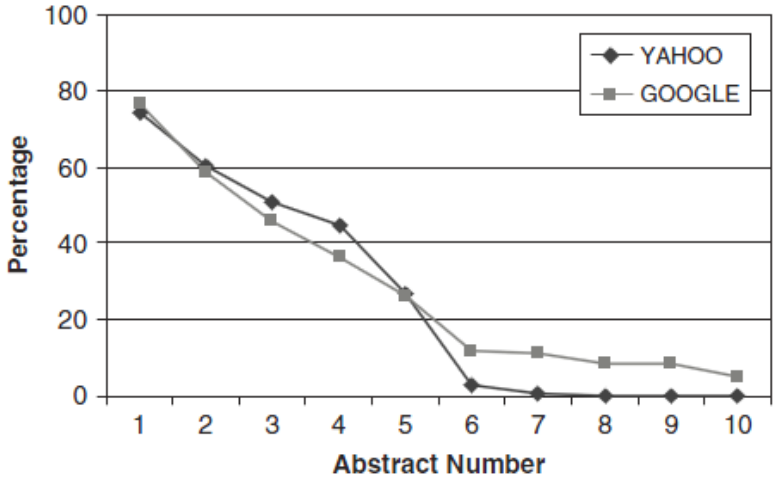
- Different task types
  - Navigational
    - Reaching a particular website
  - Informational
    - Obtaining a particular information
      - Which does not necessarily exist only on a particular site
- Users act differently depending on task type

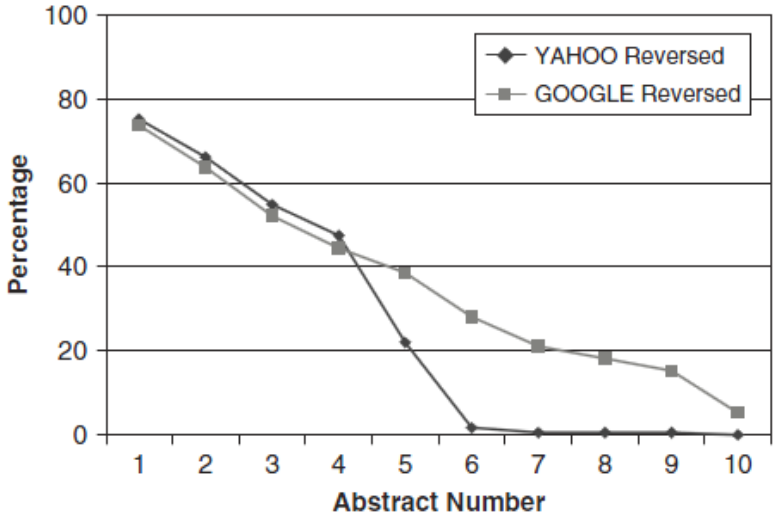
Investigating online search

- Informational vs. Navigational tasks
  - Users spend significantly more time for solving informational tasks
    - 46 seconds, compared to 34 seconds
  - For informational tasks users spend greater proportion of their time away from the results page
    - 60%, compared to 48%

Effects of the target position

- Two scenarios:
  - Normal
    - Display the results as returned by the search engine
  - Reversed
    - Display the top 10 results in reversed order
- First ranks received the most attention in both

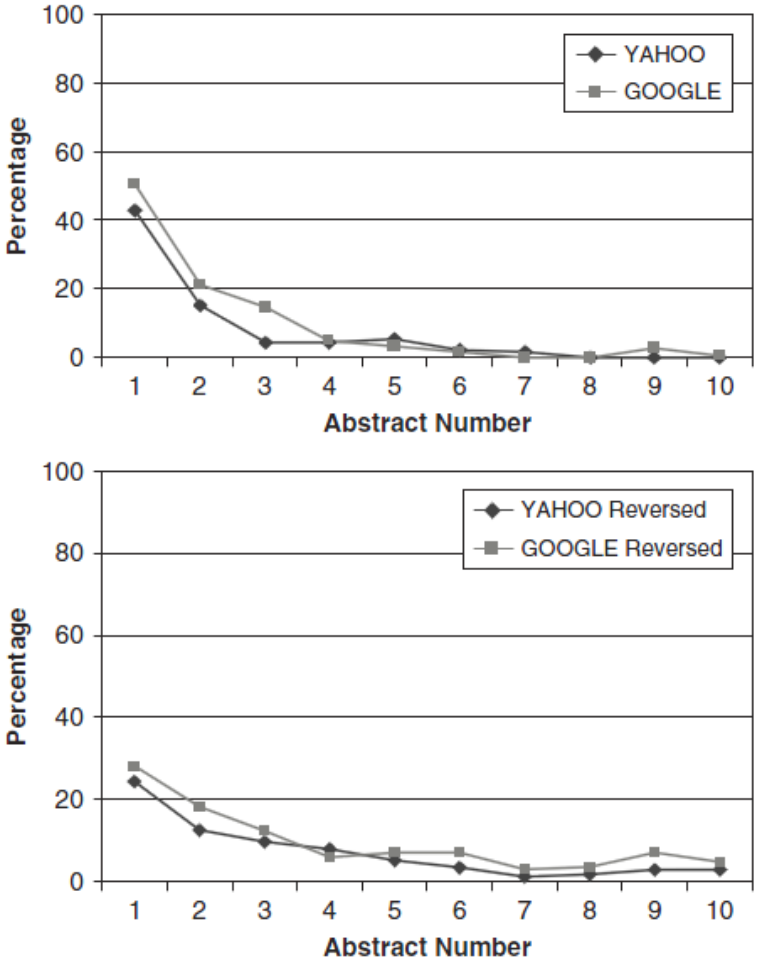




Percentage of total pages in which abstract was viewed.

Effects of the target position

- The first results received more clicks in both scenarios
- Changing results order affects success rate, increases the time on task, number of results viewed
- Users have a strong confidence in the order returned by search engines



Percentage of total pages in which abstract was clicked.

User Activity Tracking

User Activity Tracking – Goal

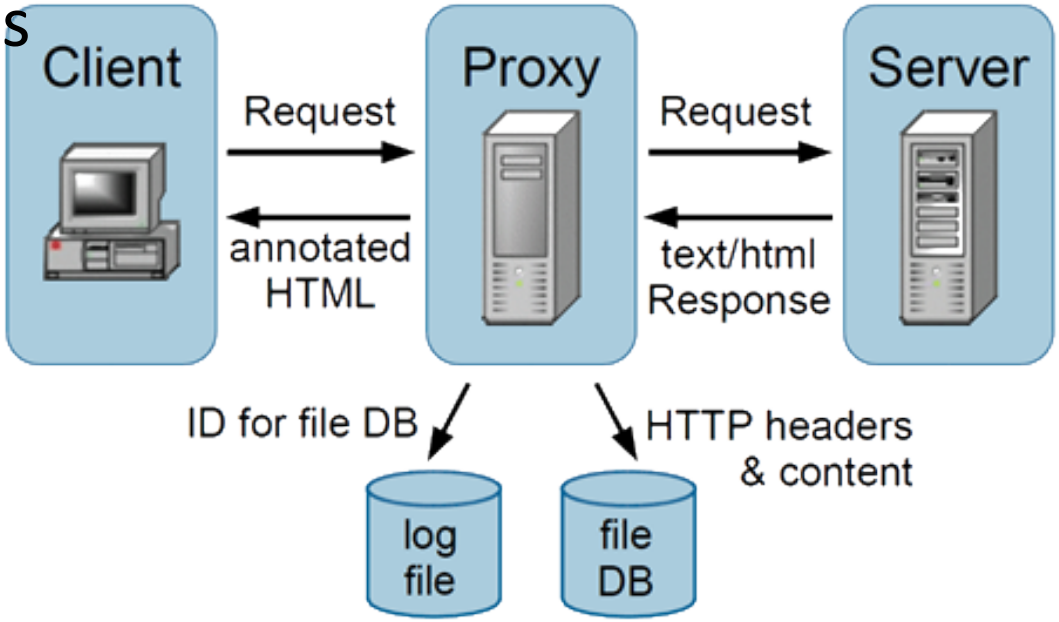
- Automate website usability tests
- Detailed logging of actions
  - mouse movements, key presses, scrolling, window resizing etc.
- Invisible tracking – unchanged user experience
- Tracking independent of website implementation

Approaches for User Activity Logging

- Client-based
  - Software installation required on client site
- Server-based
  - HTTP requests are analyzed on server
  - Problems: AJAX applications, client-side activity
- Proxy-based
  - HTTP traffic passes through an HTTP proxy

Proxy-based Logging

- The proxy modifies "text/html" responses before passing them on to the client.
- The modification causes the proxy's logging JavaScript code to be loaded by the browser.

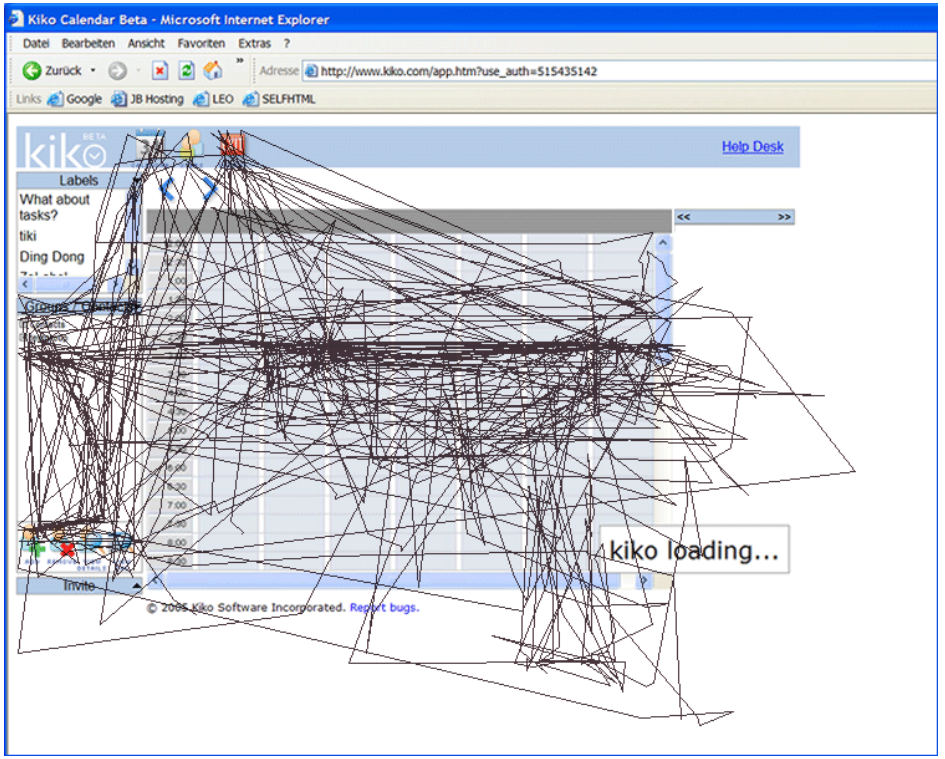


JavaScript-based Logging of User Actions

- The logging JavaScript code executes on the browser
- Registers global event handlers: *onkeypress, onmousemove, onmouseover, onfocus, onblur, onresize.*
- Log data is aggregated and passed back to the proxy at regular intervals by making HTTP requests to

Log Output

- Great level of detail for a solution without installation of client-side software
- Determine which parts of the page were viewed
- Info about click/hover coordinates and the involved DOM element



Application Areas

- Web usability tests
- User profiling/usage analysis for marketing and business process improvements
- Implicit interaction, self-adapting sites
- Developing and debugging web applications

Optimizing Search Engines using Clickthrough Data

By

Thorsten Joachims

Clickthrough Data

The set of links the user considers important

Provide specific information related to the behaviour of the user

The most important links for a specific user

Easier to acquire than user feedback (use the logs of the search engines)

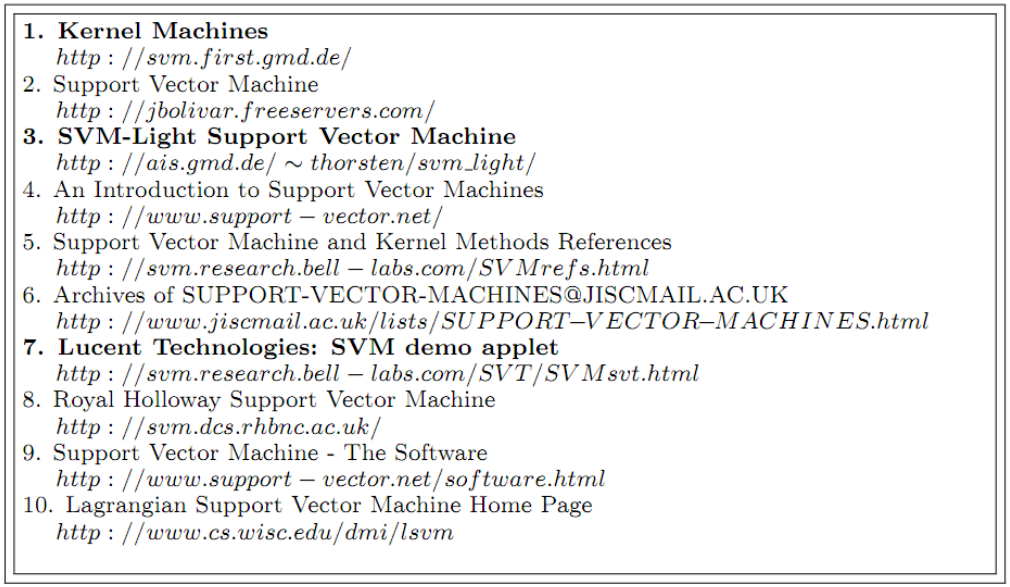
Clickthrough Data

Only the first *l (l ~ 10)* links are important

Users usually don’t consider the rest (\cite!)

For a specific user or a group of users clickthrough data could be used as training data for a ranking system

Clickthrough Data



1. Kernel Machines
http://svm.first.gmd.de/
2. Support Vector Machine
http://jbolivar.freeservers.com/
3. SVM-Light Support Vector Machine
http://ais.gmd.de/ ~ thorsten/svm_light/
4. An Introduction to Support Vector Machines
http://www.support − vector.net/
5. Support Vector Machine and Kernel Methods References
http://svm.research.bell − labs.com/SVMrefs.html
6. Archives of SUPPORT-VECTOR-MACHINES@JISCMAIL.AC.UK
http://www.jiscmail.ac.uk/lists/SUPPORT−VECTOR−MACHINES.html
7. Lucent Technologies: SVM demo applet
http://svm.research.bell − labs.com/SVT/SVMsvt.html
8. Royal Holloway Support Vector Machine
http://svm.dcs.rhbnc.ac.uk/
9. Support Vector Machine - The Software
http://www.support − vector.net/software.html
10. Lagrangian Support Vector Machine Home Page
http://www.cs.wisc.edu/dmi/lsvm

Figure 1: Ranking presented for the query "support vector machine". Marked in bold are the links the user clicked on.

	retrieval function		
	bxx	tfc	hand-tuned
avg. clickrank	6.26±1.14	6.18±1.33	6.04± 0.92

Table 1: Average clickrank for three retrieval functions ("bxx", "tfc" [23] , and a "hand-tuned" strategy that uses different weights according to HTML

Clickthrough Data

$link_3 <_{r^*} link_2$

$link_7 <_{r^*} link_2$

$link_7 <_{r^*} link_4$

$link_7 <_{r^*} link_5$

$link_7 <_{r^*} link_6$

r^*: ranking preferred by the user (binary relation)

We can generalize this preference information,

link i $<_{r^*}$ link j

for all pairs 1 <= j < i, with $i \in C$ and $j \notin C$

Learning of Retrieval Functions

Goal:

We have to find a retrival function f whose results are close to r^*

$r_{f(q)}$

In order to calculate the similarity between any given $r_{f(q)}$ and r^*, we have to use a performance metric

Average Precision (binary relevance)

Kendall's τ

Very Simple

Good Performance Metric

Learning of Retrieval Functions (2)

Kendall's τ

Between any two ranking functions the distance is,

$$\tau(r_a, r_b) = \frac{P - Q}{P + Q} = 1 - \frac{2 \cdot Q}{\binom{m}{2}}$$

D : Set of documents in a query result

P : # of concordant pairs in D x D

Q : # of discordant pairs in D x D

m : # of documents/links in D

Learning of Retrieval Functions (3)

Problem Defination of Learning an Appropriate Retrieval Function

For a fixed (but unknown) distribution of queries and target (user preferred) rankings the goal is to learn a retrieval function f(q) for which the expected Kendall's function

$$\tau p(f) = \int \tau\left(r_{f(q)}, r^*\right) d\,\Pr(q, r^*)$$

is maximal, where $\Pr(q, r^*)$ is the distribution of queries

Support Vector Machine (SVM) for Learning of Ranking Functions

Usually machine learning in information learning is based on binary classification.

(A document is either related to the query or not)

Since the information gathered from clickthrought data is not an absoulte relevancy information we cannot use binary classification

Support Vector Machine (SVM) for Learning of Ranking Functions (2)

Using a set of queries and user ranking sets (training data) we will select a ranking function among a family (F) of ranking functions that maximizes the empirical τ

$$\tau_S(\mathrm{f}) = \frac{1}{n}\sum_{i=1}^{n} \tau(\mathrm{r}_{\mathrm{f}(\mathrm{q}_i)}, \mathrm{r}_i^*).$$

Support Vector Machine (SVM) for Learning of Ranking Functions (3)

Then, we need to find a sound family of ranking functions.

How to find an F which includes an efficent ranking function (f) ?

Support Vector Machine (SVM) for Learning of Ranking Functions (4)

A set of linear rankning functions

$$(d_i, d_j) \in f_{\vec{w}}(q) \Leftrightarrow \vec{w}\Phi(q, d_i) > \vec{w}\Phi(q, d_j)$$

Where $\Phi(q,d)$ is a mapping onto features that describe the amtch betwe ent *d*

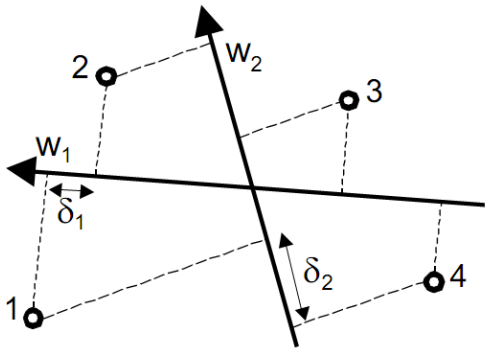


Figure 2: Example of how two weight vectors $\vec{w}_1$ and $\vec{w}_2$ rank four points.

Support Vector Machine (SVM) for Learning of Ranking Functions (5)

Instead of maximizing directly our goal function we can minimize the Q in our performance measure

$$\forall(\mathrm{di}, \mathrm{dj}) \in \mathrm{r}_1^* : \vec{\mathrm{w}} \cdot \Phi(\mathrm{q}_1, \mathrm{d}_\mathrm{i}) > \vec{\mathrm{w}} \cdot \Phi(\mathrm{q}_1, \mathrm{d}_\mathrm{j})$$

...

$$\forall(\mathrm{di}, \mathrm{dj}) \in \mathrm{r}_\mathrm{n}^* : \vec{\mathrm{w}} \cdot \Phi(\mathrm{q}_\mathrm{n}, \mathrm{d}_\mathrm{i}) > \vec{\mathrm{w}} \cdot \Phi(\mathrm{q}_\mathrm{n}, \mathrm{d}_\mathrm{j})$$

SVM^{light}

The op e

$$\begin{aligned}
&\textit{minimize:} \qquad V(\vec{w}, \vec{\xi}) = \frac{1}{2}\,\vec{w} \cdot \vec{w} + C \sum \xi_{i,j,k} \\
&\textit{subject to:} \\
&\forall(d_i, d_j) \in r_1^* : \vec{w}\Phi(q_1, d_i) \geq \vec{w}\Phi(q_1, d_j) + 1 - \xi_{i,j,1} \\
&\qquad\qquad \ldots \\
&\forall(d_i, d_j) \in r_n^* : \vec{w}\Phi(q_n, d_i) \geq \vec{w}\Phi(q_n, d_j) + 1 - \xi_{i,j,n} \\
&\forall i \forall j \forall k : \xi_{i,j,k} \geq 0
\end{aligned}$$

Experiment Setup

A meta-search engine (Striver) used for testing purposes (MSNSearch, Google, Excite, Altavista and Hotbot)

Acquires top 100 results from each search engine

Based on the learned retrival function it selects top 50 of the 500(may be lesser if more than one engine has found a specific document)

Offline Experiment

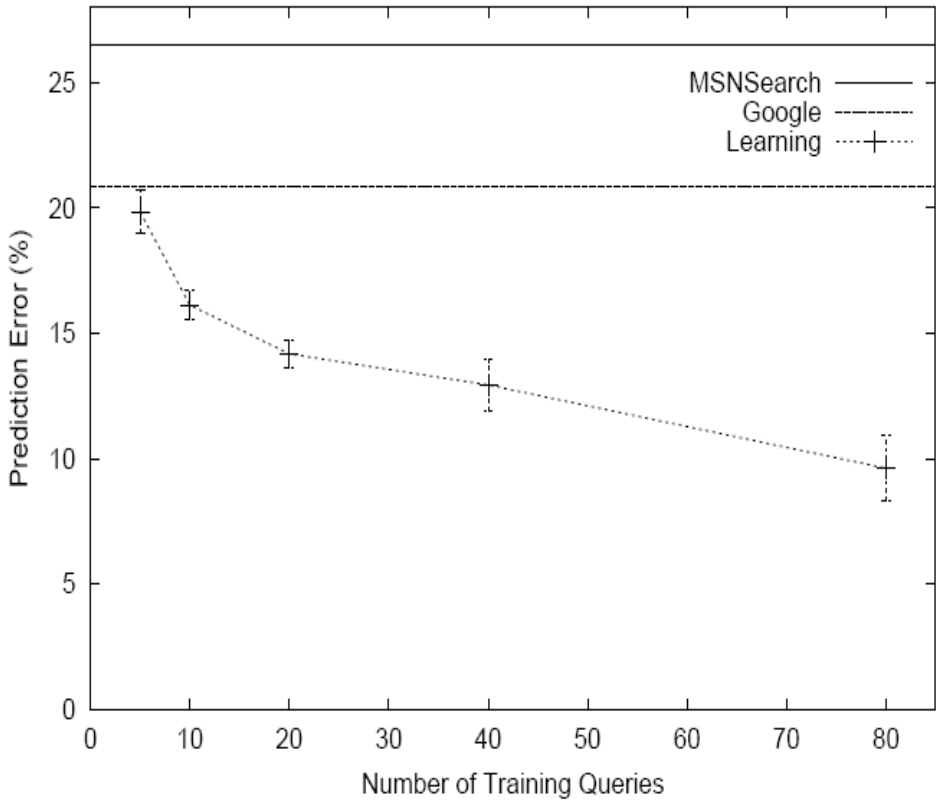
Using Striver 112 Queries are recorded

A huge set of features are used to calculate the description based retrieval functions

The testing is done with different values of training set queries

Results from Google and MSNSearch are used for benchmarking purposes

Offline Experiment (2)



Online Experiment

Striver is used by a group of people (20 people)

Based on these people's queries training set of Striver is composed of 260 queries

The results are compared with results from Google, MSNSearch and Toprank (a simple meta-search engine)

Online Experiment (2)

Comparison	more clicks on learned	less clicks on learned	tie (with clicks)	no clicks	total
Learned vs. Google	29	13	27	19	88
Learned vs. MSNSearch	18	4	7	11	40
Learned vs. Toprank	21	9	11	11	52

Table 2: Pairwise comparison of the learned retrieval function with Google, MSNSearch, and the non-learning meta-search ranking. The counts indicate for how many queries a user clicked on more links from the top of the ranking returned by the respective retrieval function.

More clicks means that (for Google) users clicked more links in the learned engine than they do in Google for 29 queries out of 88.

Less clicks means that (for Google) users clicked less links in the learned engine than they do in Google for 13 queries out of 88

Analysis of the Online Experiment

Since all of the users have used the engine for academic searches the learned data is good for searches in academic research topics

But it may not give that good results for different groups of people

We can say that learned engine is a customizable engine unlike traditional engines

(Multidocument) Text Summarization

Summarization for text documents?

- What is Summarization?

  Summary of an original text without requiring full semantic interpretation

- Tools: Word Net thesaurus, shallow parser, POS & Brill's tagger, Segmentation Algorithm

- Design steps:

  Segmentation (Create lexical chains)

  Identify Strong Chains

  Extract significant sentences

Cohesion and Coherence

- Cohesion = “sticking together”

  Achieved through the use of semantically related terms, references, ellipses and collocations

    Reiterations

    Collocations

- Lexical chains are created by using semantically related words but also between sequences of related words

- Coherence

  Macro –level semantic structure of connected discourse

Cohesion vs. Coherence

- Cohesion

  Creates connections in a non-structural manner

- Coherence

  Relations between segments such as

  - Elaboration
  - Cause
  - Explanation

Overview of the Algorithm (1)

- Segment the original text
  - •form a token sequence
  - •form a block
  - •computation of similarity
- Construct lexical chains (WordNet , POS tagger)

  select set of candidate words (noun, POS tagger)

  find appropriate chain by relations between words

  Extra-strong (same word)

  Strong (two words connected by WordNet relations)

  Medium-strong (link between synsets is longer than one)

  If found, insert the word in the chain

Overview of the Algorithm (2)

Identify strong chains

Good predictors of strength of a chain ?

Length = number of occurrences of members in a chain

Homogeneity index = 1 – (number of distinct occurrences / length)

Score(Chain) = Length * Homogeneity_Index

Strength Criterion :

Score(Chain) > Average(Scores) +

2* StandardDeviation(Scores)

Extract significant sentences ?

This is a challenge…

Overview of the Algorithm (3)

Extract significant sentences ?

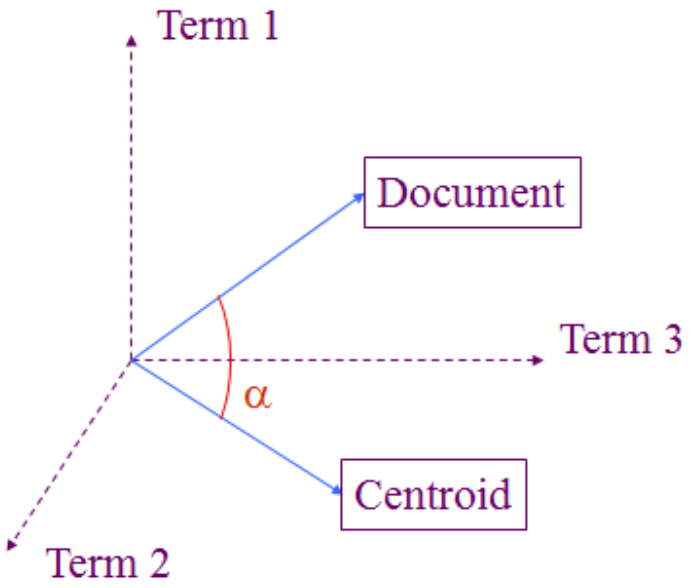
- Heuristic
  - Select representative words
  - Extract sentence with first appearance of representative Word.

MEAD Summarization

- Centroid based
- Based on sentence utility
- Vector-based representation/vector based matching
- MEAD Summarization toolkit

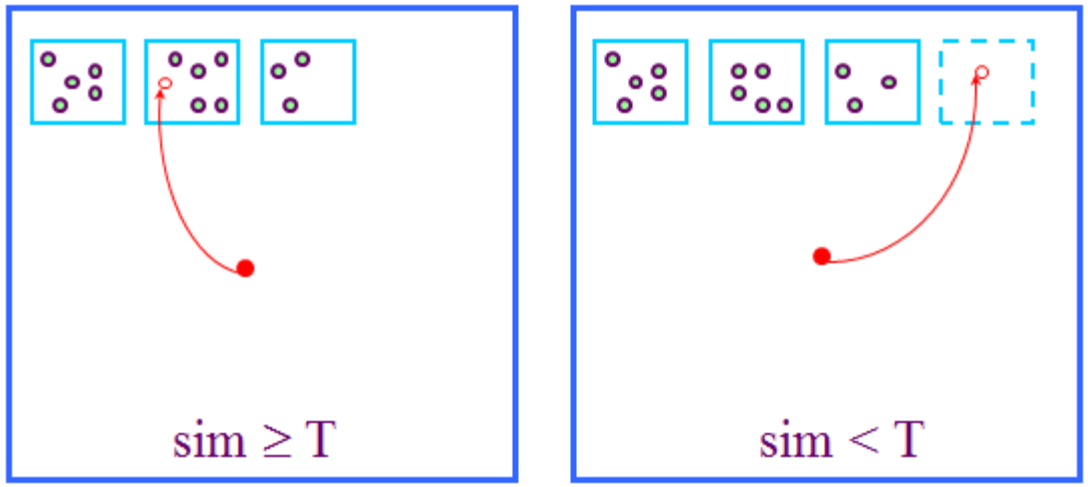
Vector-based representation

- The cosine measure



$$sim(D,C) = \frac{\sum_k d_k . c_k . idf(k)}{\sqrt{\sum_k (d_k)^2} . \sqrt{\sum_k (c_k)^2}}$$

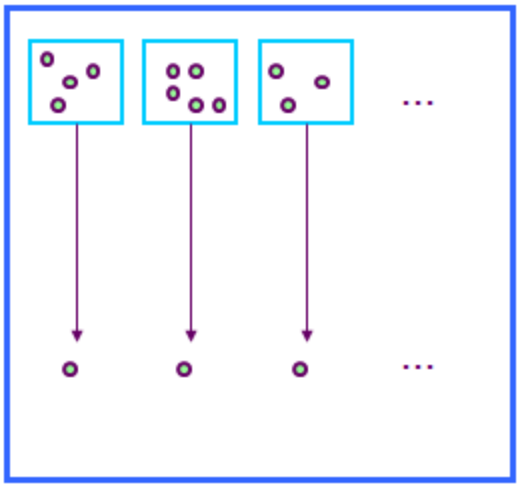
CIDR



MEAD Summarization

- Input: Cluster of d documents with n sentences (compression rate = r)
- Output: (n*r) sentences from the cluster with the highest values of score

$$SCORE\ (s) = S_i\ (w_c C_i + w_p P_i + w_f F_i)$$



MEAD Summarization (Toolkit)

- Three default features that come with the MEAD distribution are Centroid, Position and Length.
  1. Position
     - the first sentence of a document gets the maximum Position value of 1, and the last sentence gets the value 0.
  2. Length
     - Length is not a real feature score, but a cutoff value that ignores sentences shorter than the given threshold.
     - Several rerankers are implemented in MEAD
       - default reranker of the system based on Cross-Sentence Informational Subsumption(CSIS) (Radev, 2000)
  3. Centroid

MEAD Summarization (Toolkit) Policy example

which is a precomputed list of idf 's for English words.

Relative weight

The three default MEDA features

```
feature LexRank LexRank.pl 0.2
Centroid 1 Position 1 LengthCutoff 9 LexRank 1
mmr-reranker-word.pl 0.5 MEAD-cosine enidf
```

Figure 4: A sample MEAD policy.

The reranker in the example is a word-based MMR reranker with a cosine similarity threshold, 0.5

the number 9 indicates the threshold for selecting a sentence based on the number of the words in the sentence.

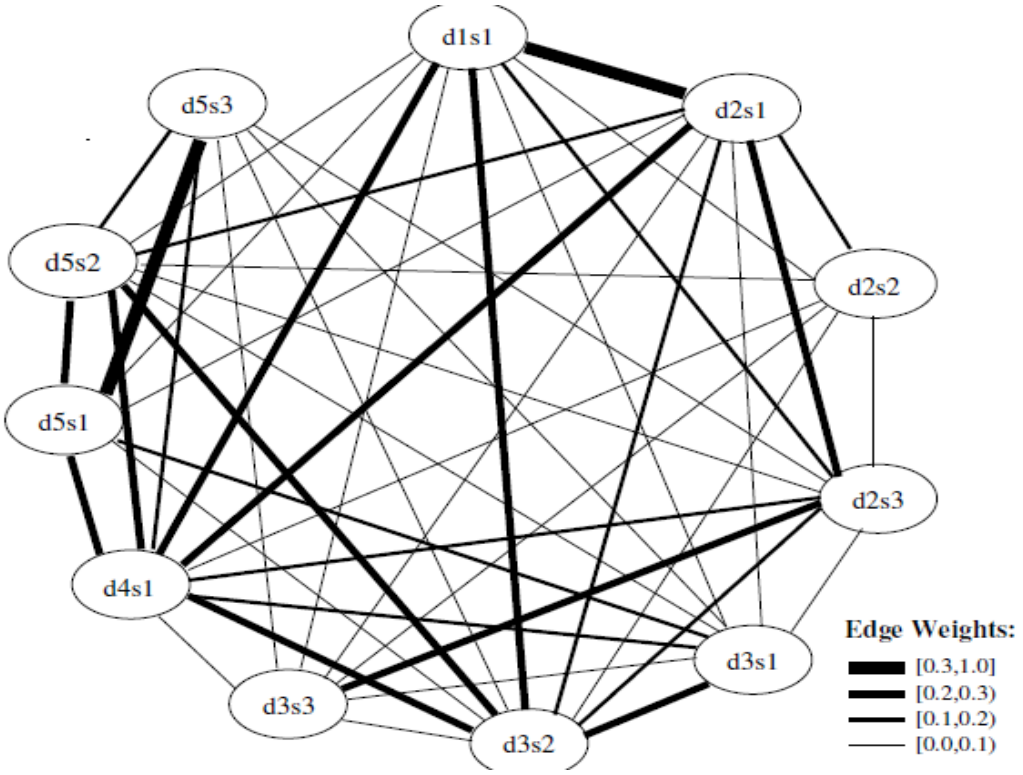
Graph-based Lexical Centrality

Cetrality based sentence salience

- Bag-of-words model → idf cosine similarity
→ cosine similarity matrix / weighted graph
→ sentence centrality

- Basic approach:
degree centrality

- Threshold variation



Eigenvector centrality (LexRank)

- Uses the concept of *prestige* from social networks
- How many friends? + Who are your friends?
- Noisy cluster with 1 unrelated document
- Each node will distribute the centrality to its neighbors

$$p(u) = \sum_{v \in adj[u]} \frac{p(v)}{deg(v)} \qquad \mathbf{p} = \mathbf{B}^{\mathrm{T}}\mathbf{p} \qquad \mathbf{B}(i,j) = \frac{\mathbf{A}(i,j)}{\sum_k \mathbf{A}(i,k)}$$

- B is a stohastic matrix
- P is the eigenvector corresponding to eigenvalue 1

LexRank

- Uses PageRank equation
- d – damping factor for convergence

$$p(u) = \frac{d}{N} + (1-d) \sum_{v \in adj[u]} \frac{p(v)}{deg(v)}$$

Variation: Continuous LexRank

- Use the weighted graph directly (no discretization)
- Normalize the similarity matrix

$$p(u) = \frac{d}{N} + (1-d) \sum_{v \in adj[u]} \frac{\text{idf-modified-cosine}(u,v)}{\sum_{z \in adj[v]} \text{idf-modified-cosine}(z,v)} p(v)$$

Performance

- Centrality is better than Centroid (subsumption, noisy data)

- Baselines for performance measurements:
  - Random sentences
  - Lead-based summarization (positioning)

- Degree centrality, LexRank with threshold, continuous LexRank are all better than baselines.

- Document Understanding Conference (DUC 2004)

Sentence Ordering

Sentence ordering

- Input → Selection → Ordering → Repairs → Summary
- Does it really matter?
- What do we use?
  - chronological ordering of events
  - topical relatedness
- Naïve algorithms:
  - Majority Ordering
  - Chronological Ordering

Majority Ordering

- Identify units of text conveying similar information → **Themes**

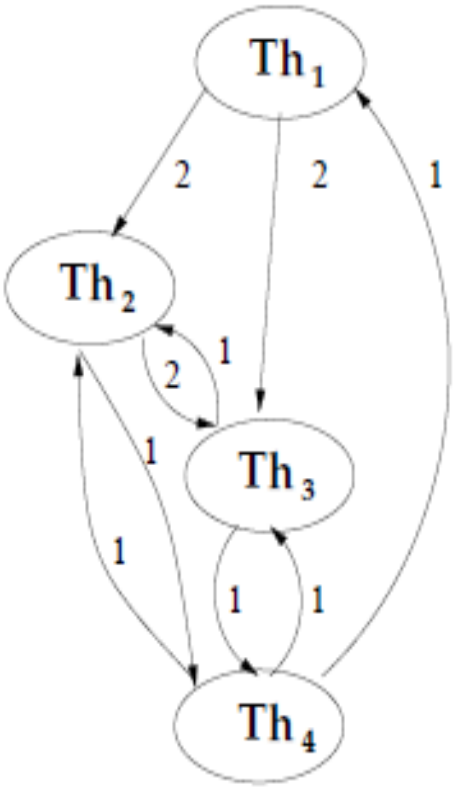
- 1 Theme → 1 sentence

- Binary relation (*Theme1 < Theme2)*

- Relation not transitive!

- Problem is NP-complete...

- … so we use heuristics



Chronological Ordering

- Also uses **themes**
- Assigns **timestamps** to themes → total order
- Explicit timestamps are not always available
- It's hard to assign timestamps to some themes (reactions to events, background information)
- Chronological approach is inherently limited

A better algorithm

- ***Majority Ordering*** - works if information is ordered consistently

- ***Chronological Ordering*** - works if information is temporally sequenced

- Idea: ***Chronological Ordering + topical relatedness***

- Group themes into cohesion blocks and then apply C. O.

Summary

- Understanding user behavior (eye tracking)
- User activity tracking
- Optimizing Search Engines using Clickthrough Data
- Using Lexical Chains for Text Summarization
- Multidocument Summarization – Centroid-based algorithm
- Graph-based Lexical Centrality
- Sentence Ordering

Questions?