I'd like to start with a cartoon.

It's about a guy who shows a cartoon before giving a boring presentation.

But it doesn't work because the cartoon has no punchline.
Motivation

- How does Google sort its results?
- Given a document and a query, which sorting is most useful?
- Ranking is based on features, such as
  - Term occurrence
  - Term proximity
  - Linguistic Features
  - Etc…
Motivation

- But, which features should we choose?
- What trade-off between cost and quality of results is optimal?
- Can we complete Ranking in a certain time?
- Common Approach:
  1. Hope
  2. Diligence
Ranking Under Temporal Constraints

Seminar on Information Retrieval
Andreas Frische, UdS
Outline

- Constrained Linear Ranking
  - Linear Ranking Functions
  - Constrained Linear Ranking
  - Algorithm: Indept
  - Feature Weight
  - Feature Cost
  - Joint Prediction Model
  - Algorithm: Joint Ranking
  - Open Questions

- Experiments
  - Experimental Setup
  - MAP vs. time
  - Satisfying Time Constraints

- Wrap Up
Linear Ranking Functions

- Many widely used Ranking Models use *Linear Ranking*
- Simple, yet effective class of ranking functions
- Given
  1. Query $q$
  2. Document $d$
  3. Features $F = f_1 \ldots f_N$ with
  4. Model Parameters $\Lambda = \lambda_1 \ldots \lambda_N$

$$Score(q, d) = \sum_i \lambda_i f_i(q, d)$$
Linear Ranking Functions

- Problem: Computational Cost is query dependent
- Example:
  - Feature Set: \{(Phrase, \lambda_P), (TF, \lambda_T)\}
  - Q1: White House
  - Q2: White House, Rose Garden

<table>
<thead>
<tr>
<th>Cost</th>
<th>Phrase</th>
<th>TF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>1 bigram</td>
<td>2 unigrams</td>
</tr>
<tr>
<td>Q2</td>
<td>3 bigrams</td>
<td>4 unigrams</td>
</tr>
</tbody>
</table>
Constrained Linear Ranking

- Basic Idea: Fill as much features into a "sack"/threshold as possible
  - Knapsack Problem
- To instantiate a model we need to
  1. Define the Cost of Features
  2. Determine the Weight of Features
  3. Select subset of features for each [class of] queries

- For now, assume we have 1 and 2 done
Algorithm: Indept

- Features are selected independently from each other

In: Time Constraint $T(q)$, Feature Set $FS(q)$, Feature Weights $\Lambda(q)$, Feature Cost $C(q)$
Out: Constrained Ranking Function $R(q)$

$R = \emptyset$, $totalcost = 0$

Compute Feature Profit Density $\forall_i p_i := \frac{\Lambda(q)}{C(f_i)}$

Queue $F :=$ Features sorted by profit density

While ($F$ not empty) Do
  Let $f$ be the Head of $F$
  Remove it
  If ($totalcost + Cost(f) < T(q)$
    Add ($f$ with $\Lambda_f$) to $R(q)$
    $totalcost = totalcost + Cost(f)$
  
Fin

Od

December 2, 2010
Feature Weight

- Feature Weights should depend on the query
  - Chocolate Milk vs Johannes Brahms

- Given
  1. Meta Features $G$
  2. Meta Feature Weights $W$

$$\lambda_i(q) := \sum_j w_j g_j(q)$$

- Example meta feature
  #times $q$ occurs in a collection, such as Wikipedia Titles
Feature Cost

- Heuristic: features with more ~Operations get a higher cost
- Weak part of the paper
- But works surprisingly well
Joint Prediction Model

- In a large Feature Set, some features may be redundant
- Solution:

  After adding a feature, *penalize* features with a similar concept
Algorithm: Joint Ranking

In: Time Constraint $T(q)$, Feature Set $FS(q)$, Meta-Features $G$, Meta-Feature Weights $W(q)$, Feature Cost $C(q)$,
Out: Constrained Ranking Function $R(q)$

$R = \emptyset, totalcost = 0$

Compute Feature Weights: $\lambda_i(q) = \sum_j w_j g_j(q)$

Compute Feature Profit Density $\forall_i p_i = \frac{\lambda_i(q)}{c(f_i)}$

Queue $F1 :=$ Features sorted by profit density, $F2 :=$ empty Queue

Group features by concept: $G_e :=$ features of concept $e$

While ($F1$ or $F2$ not empty) Do

Let $f$ be max(head $F1$, head $F2$)

Remove it

If ($totalcost + Cost(f) < T(q)$

Add ($f$ with $\lambda_f$) to $R(q)$

$totalcost = totalcost + Cost(f)$

If (concept of $f$ not covered AND $\lambda_e < \alpha$)

Reduce weight of $e$ by Redundancy Penalty

Move Features with same concept as $f$ to $F2$

Mark concept covered

$Fi$ ; Od

December 2, 2010
Open Questions

- Where do the meta-feature weights come from?
- Where does the Redundancy Penalty come from?
- Where does $\alpha$ come from?
Outline

- Constrained Linear Ranking
  - Linear Ranking Functions
  - Constrained Linear Ranking
  - Algorithm: Indept
  - Feature Weight
  - Feature Cost
  - Joint Prediction Model
  - Algorithm: Joint Ranking
  - Open Questions

- Experiments
  - Experimental Setup
  - quality vs. time
  - Satisfying Time Constraints

- Wrap Up
Experimental Setup

- We operate on the following test collections

<table>
<thead>
<tr>
<th></th>
<th>Wt10g</th>
<th>Gov2</th>
<th>Clue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topics</td>
<td>451-550</td>
<td>701-850</td>
<td>1-50</td>
</tr>
<tr>
<td># docs</td>
<td>1,692,096</td>
<td>25,205,179</td>
<td>50,220,423</td>
</tr>
<tr>
<td># docs / Topics</td>
<td>~3400</td>
<td>~33000</td>
<td>~2000000</td>
</tr>
<tr>
<td>avg qlen (title)</td>
<td>2.50</td>
<td>2.96</td>
<td>1.88</td>
</tr>
<tr>
<td>Avg qlen(desc)</td>
<td>6.08</td>
<td>5.90</td>
<td>5.88</td>
</tr>
</tbody>
</table>
Experimental Setup

- And test these Algorithms
  - ALL (features) – acts as a upper bound
  - QL (Query Likelihood) – Baseline Algorithm
  - Indep
  - Joint

- X-axis denotes time, measured in QL time cost
- Thus we become hardware independent
- Y-axis denotes quality of results
- (MAP := Mean Average Precision)
MAP vs. time

(i) Wt10g title
(ii) Gov2 title
(iii) Clue title
(iv) Wt10g description
(v) Gov2 description
(vi) Clue description

December 2, 2010
Satisfying Time Constraints

(i) Wt10g title  
(ii) Gov2 title  
(iii) Clue title  
(iv) Wt10g description  
(v) Gov2 description  
(vii) Clue description

December 2, 2010
Wrap Up

- Ranking is vital for returning useful results to a query
- Time constraints may apply
- Constraint Linear Ranking allows to construct a Ranking Function for a query and time constraint
- More time leads to better results (mostly)
Thank You

- Questions?