TREBEK
(Text REtrieval Boosted by Exterior Knowledge)

Group 6:
Chuck Curtis, Matt Hohensee, Nathan Imse
Back to the Drawing Board

- Went back and essentially re-implemented D3
- Changes to Document Retrieval:
  - Slightly more document cleaning in the indexing stage
    - Gave us slightly better MAP with 200 docs/query than we previously got with 1000 docs/query
  - Target token weights boosted to 1.5 query token weights
- Utilized Web Boosting to guide Passage Retrieval
- Utilized Thresholding of PyLucene document retrieval
  - Helped more with runtime than performance
Web Boosting

- urllib2 and BeautifulSoup python libraries
- Simple pronoun replacement for query reformulation
  - Query: When was he born?
  - Target: Fred Durst
  - New Query: When was Fred Durst born?
  - if no pronoun found, then target is concatenated to beginning of query
- Scraped result abstracts from Ask.com
  - Two settings: first page only and first 10 pages
- Why Ask.com?
  - Easy to generate URL's
  - Consistent results
Why Not Use Aranea? That's What All the Cool Kids are Doing...

- Already had most of our scraping in place before the Aranea GoPost exploded
  - didn't want to change horses mid-river
- Our scraping was plenty fast
  - essentially as fast as reading from local caches
    - 40-60 seconds for the TREC 2004 data
- No API's meant that we didn't have to worry about critical methods being deprecated
Web Boosting

- Tested the utility of web text by using it as a "passage" and computing MRR
- Attempted to reduce the average length of the web text while maintaining the MRR

<table>
<thead>
<tr>
<th></th>
<th>MRR</th>
<th>Avg # Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>First page</td>
<td>0.71</td>
<td>2413</td>
</tr>
<tr>
<td>First 10 pages</td>
<td>0.88</td>
<td>26839</td>
</tr>
</tbody>
</table>
Web Boosting -- K-Medoids

- Had no idea if it would work
- Performed K-Medoid clustering on sentences in the web text
- Cosine Similarity
- Medoids at convergence were assumed to be the more representative sentences
- Relies on repetition of answers in the web text
- Surprisingly good performance
  - not very robust against noise
Web Boosting -- Ngram Overlap...ish

- Found that unigrams were the most effective
- Each sentence in the web text was scored according to the following equation:

\[ S(x) = \sum_{i \in A} \left[ \frac{f(i, w) + f(i, x)}{\text{len}(x)} \right] + \sum_{j \in T} \left[ \frac{14 \cdot f(j, x)}{\text{len}(x)} \right] - \sum_{k \in W} \left[ \frac{f(k, x)}{\text{len}(x)} \right] \]

- \( A \) = all tokens in web text
- \( T \) = tokens in query Target
- \( W \) = question words
- \( f(i, j) \) = frequency of token \( i \) in text \( j \)
- \( w \) = web text
- \( x \) = sentence
Passage Retrieval

- D3: sentence-based algorithm
  - scored each 3-sentence window based on overlap with query terms, etc.
  - truncated if it was over 1000 characters
  - this worked reasonably well, but for D4 we want to scale to smaller windows
- Tried 2-sentence window (usually < 1000 char)
  - 0.3567 lenient MRR on first 10 question groups
- Tried extracting "most contentful" 100-char passage
  - based on NEs, titlecasing, digits, etc.
  - 0.2277 lenient on first 10 groups
Passage Retrieval Redux

- Tried using text from web boosting instead of query text
- Crawl through document looking at 1000-, 250-, and 100-char passages
  - Compute cosine similarity to web text
  - Also tried looking at passage content: boosted score slightly if passage contained titlecasing, uppercasing, or digits
  - Query text, target term, answer type not used at all
I'll take "Passage Retrieval" for $400, Alex

Results on first 10 question groups from TREC-2004:

<table>
<thead>
<tr>
<th>Window size</th>
<th>Increment</th>
<th>Lenient MRR using cosine sim only</th>
<th>Lenient MRR using cosine sim and content score</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>500</td>
<td>0.5214</td>
<td>0.5412</td>
<td>~15m</td>
</tr>
<tr>
<td>250</td>
<td>125</td>
<td>0.3804</td>
<td>0.3300</td>
<td>~18m</td>
</tr>
<tr>
<td>250</td>
<td>50</td>
<td>0.3978</td>
<td>---</td>
<td>~45m</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>0.2689</td>
<td>0.2414</td>
<td>~20m</td>
</tr>
</tbody>
</table>

Final system:
- no content scoring
- increment = half of window size
## Final Results

<table>
<thead>
<tr>
<th></th>
<th>1000 chars</th>
<th>250 chars</th>
<th>100 chars</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004 Strict</td>
<td>0.309</td>
<td>0.247</td>
<td>0.188</td>
</tr>
<tr>
<td>2004 Lenient</td>
<td>0.488</td>
<td>0.359</td>
<td>0.281</td>
</tr>
<tr>
<td>2005 Strict</td>
<td>0.243</td>
<td>0.147</td>
<td>0.117</td>
</tr>
<tr>
<td>2005 Lenient</td>
<td>0.461</td>
<td>0.273</td>
<td>0.208</td>
</tr>
</tbody>
</table>
## Improvement over D3

<table>
<thead>
<tr>
<th></th>
<th>D3</th>
<th>D4</th>
<th>% Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004 Strict</td>
<td>0.2168</td>
<td>0.309</td>
<td>+42.5%</td>
</tr>
<tr>
<td>2004 Lenient</td>
<td>0.3112</td>
<td>0.488</td>
<td>+56.8%</td>
</tr>
<tr>
<td>2005 Strict</td>
<td>0.2428</td>
<td>0.243</td>
<td>+0.1%</td>
</tr>
<tr>
<td>2005 Lenient</td>
<td>0.3795</td>
<td>0.461</td>
<td>+21.5%</td>
</tr>
</tbody>
</table>
If Only We Had More Time...

- Utilize query classification from D2 in our answer extraction
- Try things like FrameNet and Pattern Searching
- If we could get a concise answer from the web data, then we would try:
  - feeding it into our PyLucene queries
  - use more of a search than similarity-based algorithm among the documents
- Clean the TREC-related paper abstracts from the web text