LUIZ FELIX

IMPROVING HEALTH SEARCH

PROBLEM

- Users go to Search Engines in order to find more information about diseases

- It is known that major Search Engines don’t perform very well with self-diagnosis as:

- They may be imprecise, ambiguous or vague

- Users aren’t used to medical terms

WHAT WE HAVE

- A multilingual corpus of ~1,100,000 HTML pages
  ~$2.8*10^6$ tokens
  ~255 words per document, on average.

- 67 queries (66 used for evaluation)
  Smallest query: 1 word, “cavities” (?)
  Average query: 4 words
  Biggest query: 10 words

STRATEGY

- Remove the HTML tags from the corpus
  HTML tags are irrelevant and they count as tokens, which influences the Language Model calculations

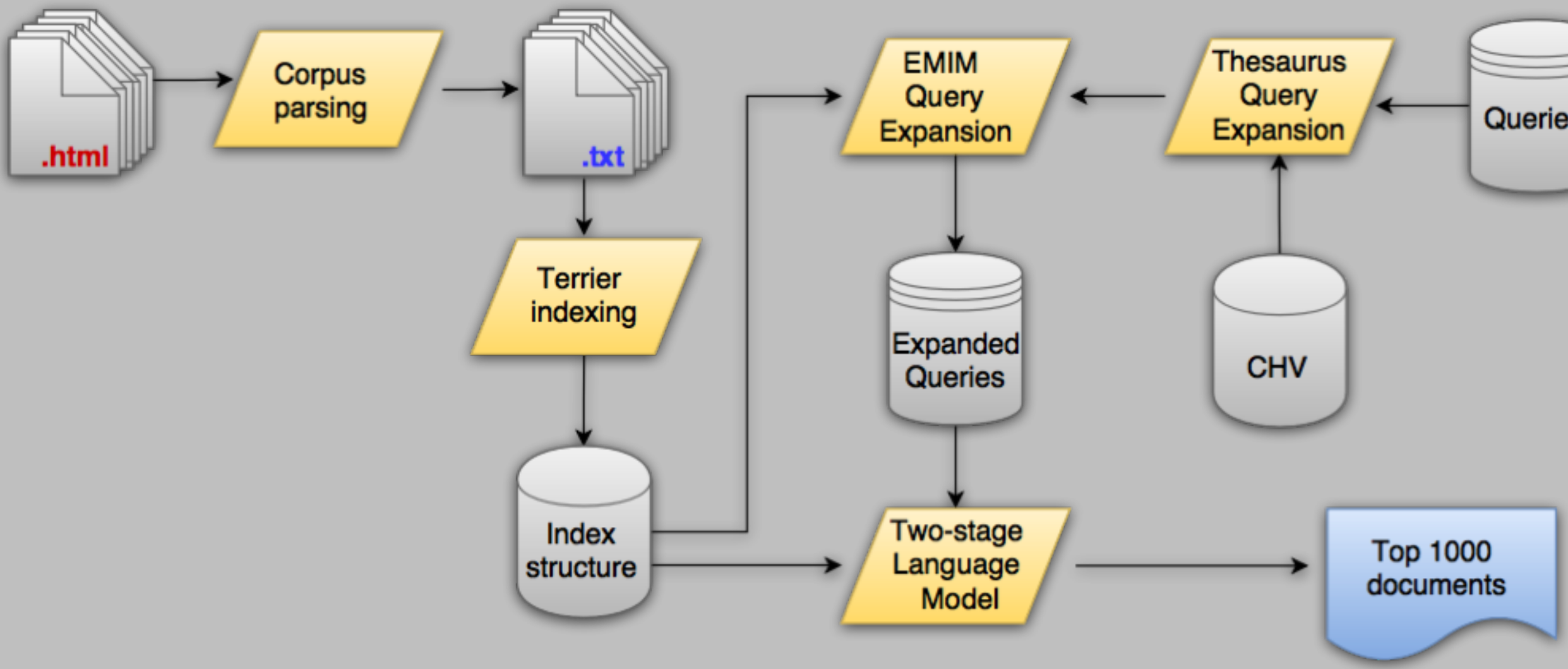
- Enhance the query with medical terms
  This may increase the diversity and relevance of retrieved documents

STRATEGY

- Expand the query by using the corpus vocabulary
  Trying to make the queries less vague

- Use a Language Model that can work both on large and small queries

STRATEGY


.html
Corpus parsing
.txt
Terrier indexing
Index structure
EMIM Query Expansion
Thesaurus Query Expansion
Queries
Expanded Queries
CHV
Two-stage Language Model
Top 1000 documents

REMOVING HTML TAGS

- Library BeautifulSoup 4 for Python was used
- Every *head, script, noscript, meta, table, ul, li, link, form, style* and *img* tags and its contents were removed
- It’s an aggressive approach, but removes everything that the user “can't see”
- Downside: content in lists and pages structured under tables are gone
- …Although lists are usually used for markup and tabled sites are from the last decade

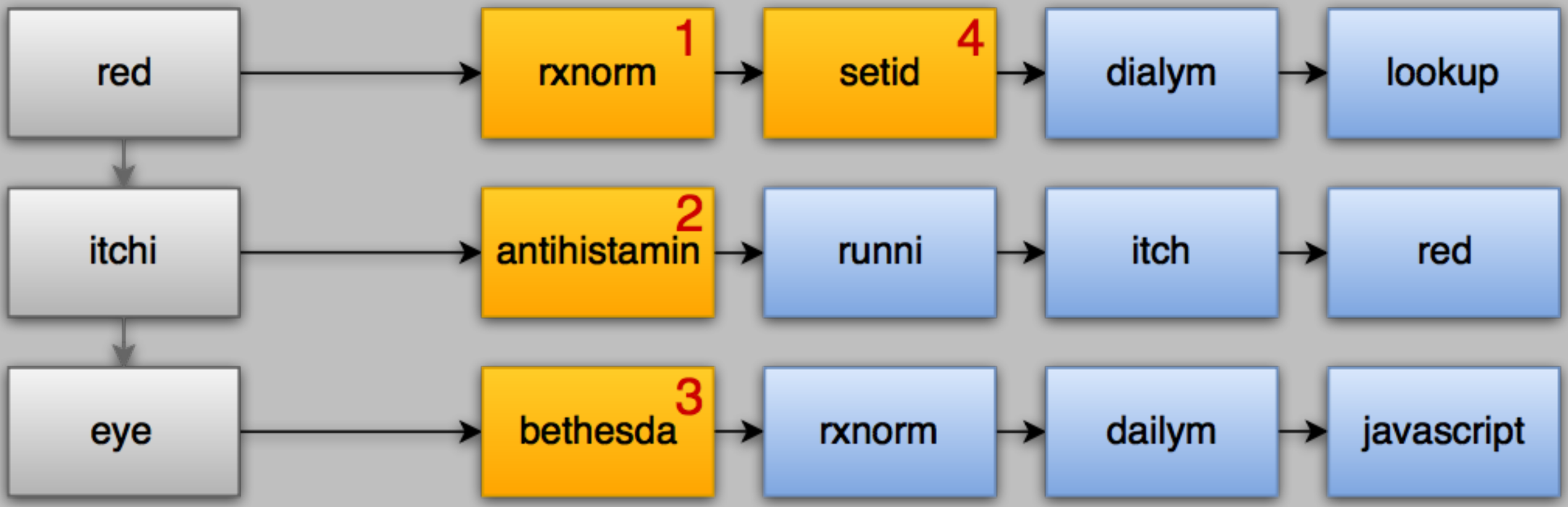
CHV QUERY EXPANSION

- The Consumer Health Vocabulary (CHV) as a “thesaurus"
- Use it to expand a lay term into a technical one
- This is done by searching for each query term on this dictionary and a combination of this term and the next
- The *k* most frequent terms are used on the expansion
- It usually yields a small (~2) amount of new terms

CORPUS QUERY EXPANSION

- First the query has the repeated and stop words removed, then it's stemmed and lowercased
- Then it is expanded based on the EMIM between two words within a document
- This is done for every term in a query
- Then the most relevant terms from each query are added on a Round-Robin fashion

ROUND-ROBIN SELECTION



Hint: red, itchy eyes are symptoms of "javascript"

EXPANSION STRATEGIES

- k = amount of words on the query
  x = amount of terms to expand

- Long: $x = max(7, 14 - k)$

- Short: $x = max(5, 10 - k)$

- Log: $x = \left\| \frac{R}{log(1+k)} \right\|$

TWO-STAGE LANGUAGE MODEL

- Combines both Dirichlet (good for long queries) and JM (good of small queries) models

$$p(q_i|d_j) = \prod_{i=1}^{n} \left[(1-\lambda) \frac{c(q_i, d_j) + \mu p(q_i|C)}{|d_j| + \mu} + \lambda p(q_i|U) \right]$$

- To avoid precision errors, log probabilities were used

$$log[p(q_i|d_j)] = \sum_{i=1}^{n} log \left[(1-\lambda) \frac{c(q_i, d_j) + \mu p(q_i|C)}{|d_j| + \mu} + \lambda p(q_i|U) + 2 \right]$$

ESTIMATING μ

- The estimation is done by using the following function

$$\mu = argmax_{\hat{\mu}} \sum_{i=1}^{N} \sum_{w \in V} log \left[\frac{c(w, d_i) - 1 + \hat{\mu} p(w|C)}{|d_i| - 1 + \hat{\mu}} \right]$$

- Which can be maximised though Newton's Method

$$\hat{\mu}^{(k+1)} = \hat{\mu}^{(k)} - \frac{g(\hat{\mu}^{(k)})}{g'(\hat{\mu}^{(k)})}$$

- *g(•)* is the derivative of the target function. Won't discuss it, as it adds nothing to the project

ESTIMATING λ

- This estimation depends on the queries, and not on the collection
- It needs the direct index to do it
- Couldn’t find a way to do it using Terrier
- For curiosity this is half of the formula (not as hard as it seems)

$$\lambda^{(k+1)} = \frac{1}{n} \sum_{i=1}^{N} \pi_i^{(k+1)} \sum_{j=1}^{n} \frac{\lambda^{(k)} p(q_j|U)}{(1-\lambda^{(k)}) p(q_j|d_i) + \lambda^{(k)} p(q_j|U)}$$

RESULTS

Query Expansion	μ	λ	P@10
—	2500	0	0.2091
—	303	0	0.2227
—	303	0.2	0.2227
—	303	0.9	0.2212
CHV	2500	0	0.1591
CHV	303	0	0.1636
Short	2500	0	0.0470
Short	303	0	0.0455
Long	2500	0	0.0273
Long	303	0	0.0258
Log	2500	0	0.1152
Log	303	0	0.1212

BUT... WHAT WENT WRONG?

- Maybe the Round-Robin strategy used to append elements to the query isn’t good
- Maybe a "sliding windows” approach works better
- Also, let’s take a look on the log-DJM formula again

$$log[p(q_i|d_j)] = \sum_{i=1}^{n} log\left[(1-\lambda)\frac{c(q_i,d_j)+\mu p(q_i|C)}{|d_j|+\mu} + \lambda p(q_i|U) + 2\right]$$

- Since we don’t have a query log, the background probability was calculated based on the collection data

BUT... WHAT WENT WRONG?

- Then $p(q_i|U) \approx p(q_i|C) = \frac{f_{q_i,C}}{|C|}$
- And we know that |C| ≈ 2.8*10^6
- The probability of even the most frequent term is too small
- Also, *log(x+y)* ≈ *log(x)*, if *x* ≫ *y*, causing:

$$log[(1-\lambda)D + \lambda J + s] \approx log[(1-\lambda)D + s]$$

- Thus justifying why **λ** had no effect over the retrieval

QUESTIONS?