Health Search Engine Project Documentation

Harry Scells

25 October 2015

github.com/hscells/health-search

1 Methods Used for Retrieval

The methods used in this system to ensure effective retrieval for the task include the following:

- Query Expansion
  - Tests were done to see which expansion model was better using two models:
    - Expected Mutual Information Probability (emim)
    - Bose-Einstein 1 (bo1) [1]
    - $w(t) = tfc \cdot log_2 \frac{1+P_n(t)}{P_n(t)} + log_2(q + P_n(t))$
    - $P_n(t) = \frac{tfc}{n}$
- Dictionary Lookup (using the Consumer Health Vocabulary)
- Query Term Weighting (qw)
  - Modified Concept Weighting algorithm [2]
  - $p(c_i|q) = \frac{h_k(c_i)}{\sum_{c_i \in q} k_h(c_i)}$
  - $h_k(c_i) = log\left[1 + \frac{c_i}{n}\right]$
  - where n is the number of terms in the top t documents, q is the query and c_i is the term
- Query term weighting was planned to be a novel implementation of another language model, however it resulted in a less than ideal accuracy.
  - This could possibly be due to the fact that the incorrect elastic search query is being used, but the alternative was the boolean model. Elastic search does not provide an interface for weighting/boosting individual terms in a query other than the way implemented

The Values that can be tuned in this system are the following:

- Standard BM25 values, b = 0.75, k1 = 0.5
- Query expansion values
  - n = number of documents to use for expansion
  - t = number of terms to use for expansion

bo1 was chosen over emim because

- It takes into account the collection as a whole, as well as the top n documents
- Generally, it expands queries with more relevant terms
- emim only looks at the position of the initial query terms in the top n documents, bo1 will look at each term in the top n documents and it's probability occurring in them as well as it's overall "importance" in the rest of the collection
  - For technical/medical terms this is desirable

It was hoped that the combination of these different models and algorithms would provide a system which would:

- Expand a query given an input query
- Weight terms in the query based on some relevance metric
- Return a list of documents which correctly diagnose a set of symptoms

The problems faced were that:

- Query expansion was very slow and the baseline outperformed it
- Weighting/boosting individual terms gave even worse results, even without query expansion

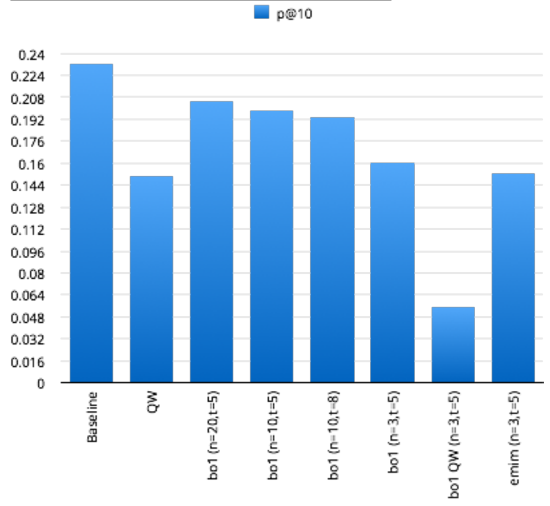
[1] UBML participation to CLEF eHealth IR challenge 2015: Task 2

[2] Discovering Key Concepts in Verbose Queries

2 Evaluation of the Results

The following table shows the p@10 results for each model

Model	p@10
baseline	0.2323
bo1(n=20,t=5)	0.2046
bo1(n=10,t=5)	0.1985
bo1(n=10,t=8)	0.1938
bo1 (n=3,t=5)	0.1600
emim (n=3,t=5)	0.1523
qw	0.1508
bo1 (n=3,t=5) + qw	0.0554



- It can be seen that by also combining models (such as that of query weighting and Bose-Einstein 1), the accuracy took a dramatic drop in performance.
- The more documents used to expand the query, the better the results
- A number higher than five for the t value will start to include even more terms which are no longer relevant
- Although query expansion and query term weighting should provide higher accuracy, it was found that the default Lucene search could not be beaten.
- Through the combination of steps listed in the previous secion, it was predicted that after the query had been expanded there would be nonsense terms added. Through weighting and boosting individual terms accordingly, the ideal outcome would be that terms which were not relevant to the diagnosis would have a lower weighting and thus not be boosted much or at all
  - This was found to be incorrect, and the models used were obviously not suited well enough to the task

3 Effectiveness of the System

- The effectiveness of the system is quite poor.
- The main problem faced with the results was the fact that all of the models treat elastic search as a black box
  - there is no way to see what elastic search does with the data
  - all communication and operations on the index must happen via a REST API
- Index mapping may not be being applied. For instance take the following query expansion: cloudy pupil affecting vision $\rightarrow$ affecting cloudy vision pupil cataracts conditions surgery **eyes eye**.
  - The two terms "eye" and "eyes" are added, even though snowballing and Lucene's default analysers are explicitly used.
- Taking the black-box approach allowed models to be represented without the need to know about the internal system, however
  - the performance cost suffers massively
  - it is impossible to modify any internal algorithms for testing purposes, and;
  - the inputs to the system are limited to what the system allows to be inputted (types of queries etc.)
- The variation of terms that get expanded is higher than expected:

  - Most queries contain a term relating to the diagnosis: baby cough $\rightarrow$ cough baby whooping medicines adults topics healthy
  - Some queries seem to diagnose the wrong problem(s): raised red lumps skin $\rightarrow$ skin lumps red raised problems cysts cellulitis baby psoriasis
  - Less queries have terrible expansion: scaly rash $\rightarrow$ scaly rash vertebrate inverterbrate allergy diagnose infection
  - When irrelevant terms are added to the query it is most likely that this impacts the search in a dramatic way

- To get better results
  - Improve the query expansion algorithm, so that it outperforms the baseline
  - Create a metric for weighting terms in a query which boost terms higher the more relevant they are
  - Do these things within the search engine itself, rather than treating it like a black box