Ancestry.com Online Forum Test Collection

Jonathan L. Elsas (CMU)

CMU-LTI-017

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

The Ancestry.com Online Forum Test Collection

Jonathan L. Elsas (jelsas@cs.cmu.edu)

October 5, 2011

This report outlines the construction of the Ancestry.com Forum document collection and information retrieval test collection. The Ancestry.com Forum Dataset was created with the cooperation of Ancestry.com in an effort to promote research on information retrieval, language technologies, and social network analysis. It contains a full snapshot of the Ancestry.com online forum, boards.ancestry.com, from July 2010. This message board is large, with over 22 million messages, over 3.5 million authors, and active participation for over ten years. In addition to the document collection, queries from Ancestry.com's query log and pairwise preference relevance judgements for a message thread retrieval task using this online forum are distributed.

This report is split into two parts: First we describe the dataset contributed by Ancestry.com, including the document collection and query set. Next, we describe the building of an information retrieval test collection for thread search.

Acknowledgements: This collection would not have been possible without the generosity and support of Ancestry.com, and specifically Lee Jensen who oversaw the data collection and assessment. Extensive discussions with Ben Carterette on test collection development were invaluable in the creation of the information retrieval test collection.

Part I
Document Collection and Query Set

1 Document Collection

The Ancestry.com document collection was gathered in collaboration with the company. Ancestry.com is an online resource for historical and genealogical research. The associated forum provides users of the site with a mechanism to discuss their research, share findings, and collaborate. The online forum has been active for well over ten years, and contains more than 22 million messages. This dataset is organized hierarchically with messages organized into threads, and threads organized into subforums. A full copy of the messages database was generated in July of 2010, containing all messages accessible on the forum at that time. Table 1 gives dataset size statistics.

Number of Messages	22,054,728
Number of Threads	9,040,958
Number of Sub-forums	165,358
Number of Unique Users	3,775,670
Message Date Range	December 1995 - July 2010
Mean (Standard Deviation) Message Length	94.0 (155.9) tokens
Mean (Standard Deviation) Thread Size	2.4 (4.8) messages

Table 1: Ancestry.com Forum Dataset Statistics.

1.1 Document and Thread Structure

The following metadata is included with the messages:

- Unique message identifier (`DOCNO`)
- Unique message identifier, assigned by Ancestry.com (`PID`)
- Subforum name (`SUBFORUM`)
- Thread identifier, unique per subforum (`THREAD_ID`)
- Post identifier, unique per subforum (`POST_ID`)
- Author name (`AUTHOR_NAME`)
- Unique numeric author identifier (or 0 if missing) (`AUTHOR_NAME`)
- Publication date (or 01-01-1900 if missing) (`AUTHOR`)
- URL of the original document (`POST_URL`)
- Message title (`POST_TITLE`)
- Message body (`TEXT`)

Note that some of the messages have missing metadata fields: 102,219 messages (0.46%) have no author identified, and 279 (0.001%) messages have no date specified. The message documents are distributed in `trectext` format, and an example message is shown below

```
<DOC>
<DOCNO>localities.northam.usa.states.kentucky.unknown.0004W8.003266314</DOCNO>
<PID>003266314</PID>
<SUBFORUM>localities.northam.usa.states.kentucky.unknown</SUBFORUM>
<DATE_STR>24 Feb 2001</DATE_STR>
<DATE_NUM>200102241200</DATE_NUM>
<THREAD_ID>0004W8</THREAD_ID>
<POST_ID>0004W8</POST_ID>
<POST_URL>http://boards.ancestry.com/localities.northam.usa.states.kentucky.unknown/6344/mb.ashx</POS
<AUTHOR_NAME>Patricia</AUTHOR_NAME>
<AUTHOR>625758</AUTHOR>
<POST_TITLE>Born as KRISTA DAWN MAHAR, 12-16-70, Memphis Tn,
need medical information</POST_TITLE>
<TEXT>My sister is searching for any Mahar relatives. She
needs medical information only please help.</TEXT>
</DOC>
```

The DOCNO field is a concatenation of the SUBFORUM, THREAD_ID and PID fields. The POST_ID and THREAD_ID fields can be used to reconstruct message threading information as follows:

1. Messages belonging to the same thread have identical SUBFORUM and THREAD_ID values.
2. The first message of a thread has matching THREAD_ID and POST_ID values, as in the example above.
3. A message (A) is a direct response to another message (B) in the thread if message A's POST_ID contains message B's POST_ID followed by four characters.

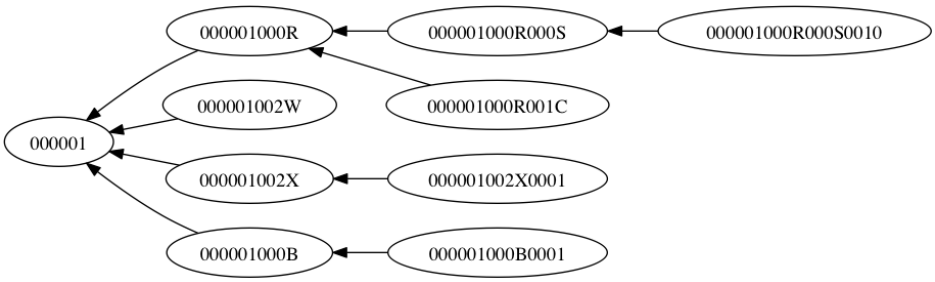

Figure 1: Threading structure given by the content of the POST_ID fields. POST_ID of the first message in thread shown at left, and arrows show message response relationship.

Figure 1 shows a sample threading structure based on the content of the POST_ID fields. Figure 2 shows the user and message volume over time. Figure 3 shows the message and thread volume distributions.

1.2 Subforum Structure

The subforum hierarchy in the Ancestry.com Forum is organized in three primary dimensions. First, a location-based subforum organization provides a mechanism for authors to associate a geographical location with their threads. Second, a surname-based subforum organization allows authors to associate their threads with an associated surname. Third, a smaller topical organization exists, with topics such as religion, military service and adoption. Several example nodes of the hierarchy are shown below:

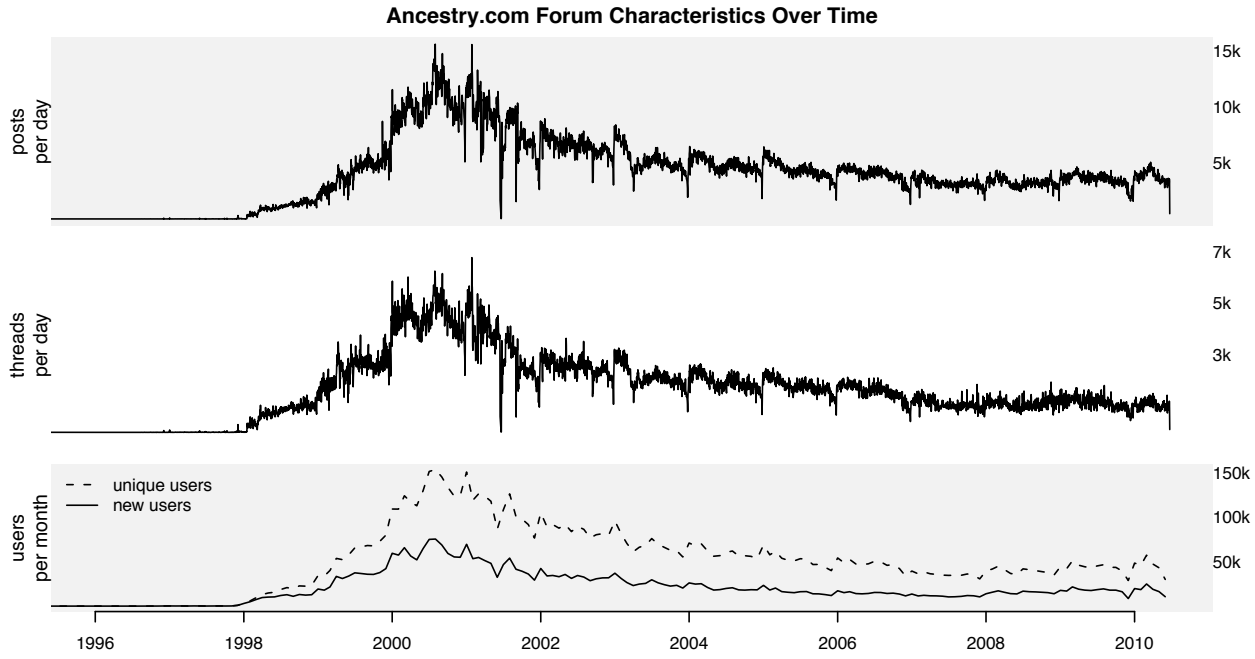

Figure 2: Ancestry.com forum characteristics over time. From top to bottom: Daily message volume, Daily thread volume, Monthly user volume. Increase in forum activity around 2000 corresponds to Ancestry.com's purchase of RootsWeb (www.rootsweb.ancestry.com), inheriting its online forum and combining the two sites' user bases. The steady decline in forum activity after the peak around 2001 reflects Ancestry.com's focus on providing more genealogical content and site functionality outside of the online forum, and users' decreased reliance on the online forum for performing research. Periodic annual volume dips correspond to December holidays.

```
Localities > North America > Canada > British Columbia > Kootenay
Localities > Western Europe > Belgium > Brussels
Localities > Middle East > Kuwait > General
Surnames > Abigail
Surnames > Efman
Surnames > Graal
Topics > Cemeteries & Tombstones > Europe > France
Topics > Military > American Revolution > Delaware
Topics > Photographs > Vintage Photograph Identification
```

Root node	Num. Subforums	Num. Threads	Num. Messages
`Localities`	4632 (2%)	4098253 (45%)	9151570 (41%)
`Surnames`	158350 (96%)	4409776 (48%)	11840781 (53%)
`Topics`	1550 (1%)	461472 (5%)	870326 (4%)
other	826 (0.5%)	74234 (0.8%)	192051 (0.8%)

Table 2: Ancestry.com subforum sizes, showing number of subforums, threads and messages under the largest three top nodes. Percent of total collection shown.

Table 2 shows the sizes of these largest three subforum organizations in the collection. Although there are many more `surnames` subforums, they tend to contain many fewer messages and threads. The `localities` and

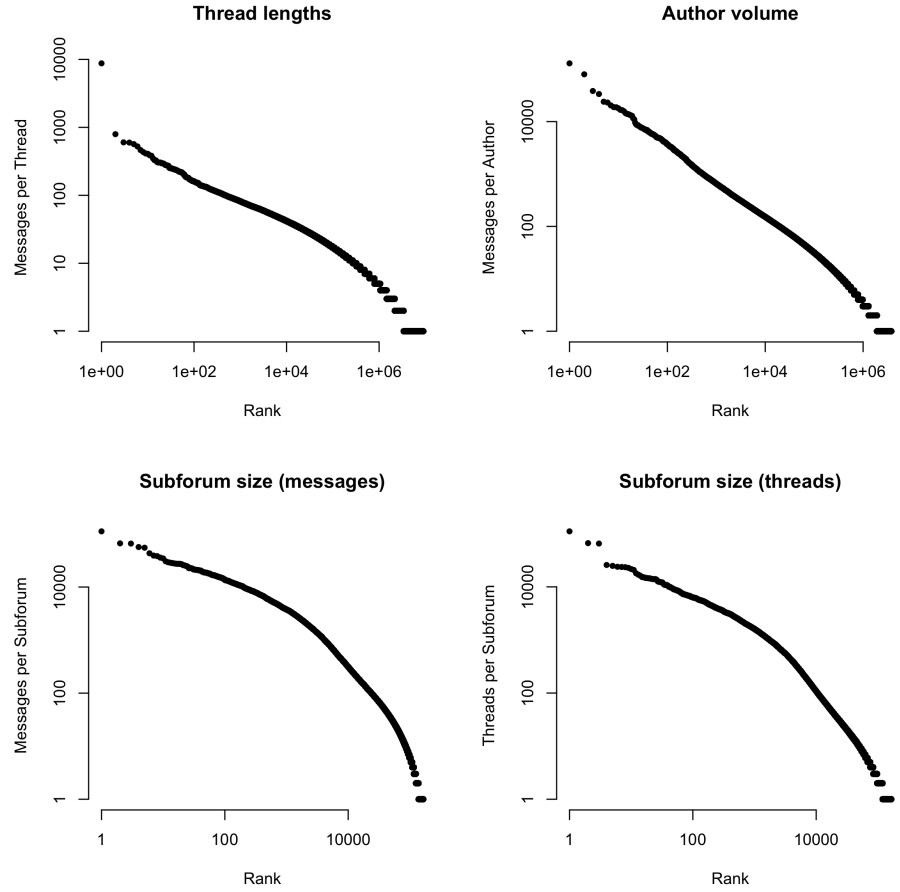

Figure 3: Ancestry.com Forum size distributions. Top: thread length and author volume distributions. Bottom: subforum size distributions (messages & threads).

`surnames` each account for slightly less than half the entire online forum thread volume. This primary organization into both a geographic and surname listing occasionally leads to an author posting a question in multiple subforums, referring to two aspects of their question. Although it leads to some redundancy in the collection, this type of organization does correspond to the primary ways users access the data, as we will show below when discussing the query set.

2 Query Set

In addition to the document collection, we present an analysis of a set of 10,000 search queries likely to have relevant documents in this collection. The search queries are extracted from Ancestry.com's primary search engine query log. Although this search engine retrieves documents from across the site, not just limited to the online forum, the queries are likely to be representative of the type of information needs that could be served by the online forum documents.

The queries issued to the main Ancestry.com search engine[1] are structured queries, containing different fields for names, locations, dates, or other fields present in their data. Figure 4 shows the internal representation of a typical query entered on the main Ancestry.com search engine. Many of these fields refer to the rich annotations generated by

[1] `http://search.ancestry.com/search/`

```
(rank
  (given-name
    (weight 100 (field 80004002 martha))
    (weight 90 (field 80000002 martha)))
  (surname
    (weight 100 (field 80004003 kekahuna))
    (weight 90 (field 80000003 kekahuna)))
  (secondary-given-name
    (weight 40 (field 80014002 yvonne)))
  (place 500
    (location 82004010 (place-text oahu, hi))
    (location 82004220 (place-text oahu, hi))
    (weight 80 (location 82004000 (place-text oahu, hi)))
    (weight 60 (location 82000000 (place-text oahu, hi)))))
```

Figure 4: Ancestry.com sample structured query.

Field Type	Ancestry.com query fields	Query Frequency
Name	surname	9860
	given-name	9112
	secondary-surname	3864
	secondary-given-name	4255
Location	place	6970 (4642 usable)
Other Keywords	keyword	560 (504 usable)
Date	date	6923

Table 3: Ancestry.com query set analysis. Note: some field values refer to numeric location identifiers, or other information not usable by our system. These values are ignored for our purposes.

Ancestry.com on some parts of their document collection, such as census or military records which may have locations, dates and names identified. The different structural fields present in the data are visible in the query, as well as weights placed on those fields in Ancestry.com's internal ranking algorithm. Ancestry.com does not annotate those fields in the online forum documents, and for this reason, we convert the structured queries to keyword queries by extracting only textual information. The query shown in Figure 4 would be converted to the keyword query *[martha kekahuna yvonne oahu hi]*. All attempts were made to retain a sensible ordering of query terms, so that order and proximity query operators continue to be effective.

Although we do not directly use the fields in the Ancestry.com queries, we can inspect the query structure to learn about the types of information needs users of the system typically have. Table 3 shows an analysis of the 10,000 queries in our query set, and Figure 5 shows the frequency of field combinations in the query set. The vast majority of queries (99%) contain a name, and approximately half (49%) contain a name and some other information, either a location or keywords. Although the date field is relatively common, the online forums collection does not have this field annotated. Additionally, it is unlikely that date information would be as useful in the forum collection rather than other documents such as census or military records where publication dates are a primary organizational facet. We sample queries for assessment to conform to this observed distribution, with 97 containing only a person name, and 94 containing a name as well as additional query terms such as location or keyword. We refer to these sets of queries as the *name* and *name+* queries respectively. Table 4 shows the query length statistics.

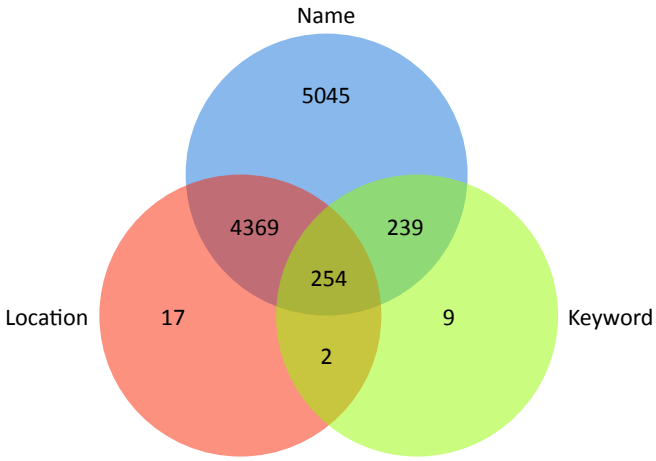

Figure 5: Ancestry.com Query Field Frequency. Frequency of different query fields in the sample of 10,000 Ancestry.com search queries. Note: only usable field values included in counts.

	Query Length			
	1	2	3	4
Number *name* Queries	10	52	32	3
	Query Length			
	3-4	5-6	7-8	9-14
Number *name*+ Queries	15	26	23	30

Table 4: Ancestry.com query length statistics.

Part II
Thread Search Information Retrieval Test Collection

3 Document Pooling

This section describes the process of pooling for identifying those documents to judge with respect to each query. The pooling process has two steps: First, a set of diverse retrieval systems must produce retrieval runs, or document rankings for each query. Second, using those retrieval runs, the most likely relevant documents must be selected for assessment.

3.1 Retrieval Runs

Typically in TREC evaluations (for example the Blog track [16, 11, 17]), many different teams with different underlying retrieval systems submit a variety of retrieval runs. With a diverse set of underlying systems and a diverse set of techniques, the resulting rankings are likely to contain a sufficient diversity for a reliable evaluation.

In the absence of a set of runs contributed by different TREC participants, we must simulate a document pool. Our goal in the creation of retrieval runs is as follows:

- First, we aim to diversify the underlying system software in order to avoid overly biasing results towards one indexing or ranking method.
- Second, we aim to include *highly effective* runs and select systems and algorithms that have a history of performing well at TREC.
- Third, we aim to include diverse features of ranking algorithms. For example, we include runs that use only a bag-of-words retrieval model, as well as runs that include fielded retrieval models.

For our retrieval runs, we use four different retrieval system in a variety of configurations, for a total of eight different message ranking algorithms. The systems and message retrieval algorithms used are listed below:

- **Indri**[2] using a variety of query formulations:
  - **Bag-of-words** queries, eg.

```
#combine(john stephen manley)
```

  - **Dependence Model** (DM) queries [13], using the suggested model weights, eg.

```
#weight(
  0.8 #combine(john stephen manley)
  0.1 #combine(#1(john stephen) #1(john stephen manley)
              #1(stephen manley))
  0.1 #combine(#uw4(john stephen) #uw4(john manley)
               #uw4(stephen manley) #uw8(john stephen manley)))
```

  - **Fielded query** with **linear combination**, eg.

```
#wsum(
  0.4 #weight(
    0.8 #combine( john.(post_title) stephen.(post_title) manley.(post_title) )
    0.1 #combine( #1( john stephen ).(post_title)
                  #1( john stephen manley ).(post_title)
                  #1( stephen manley ).(post_title) )
    0.1 #combine( #uw4( john stephen ).(post_title)
                  #uw4( john manley ).(post_title)
```

[2] Indri version 2.12 (Lemur version 4.12), `http://lemurproject.org/`

```
                #uw4( stephen manley ).(post_title)
                #uw8( john stephen manley ).(post_title) ) )
0.4 #weight(
  0.8 #combine( john.(text) stephen.(text) manley.(text) )
  0.1 #combine( #1( john stephen ).(text)
                #1( john stephen manley ).(text)
                #1( stephen manley ).(text) )
  0.1 #combine( #uw4( john stephen ).(text)
                #uw4( john manley ).(text)
                #uw4( stephen manley ).(text)
                #uw8( john stephen manley ).(text) ) )
0.2 #weight(
  0.8 #combine( john.(subforum) stephen.(subforum) manley.(subforum) )
  0.1 #combine( #1( john stephen ).(subforum)
                #1( john stephen manley ).(subforum)
                #1( stephen manley ).(subforum) )
  0.1 #combine( #uw4( john stephen ).(subforum)
                #uw4( john manley ).(subforum)
                #uw4( stephen manley ).(subforum)
                #uw8( john stephen manley ).(subforum) ) ) )
```

  - **Fielded query** with **loglinear combination**, similar to the above formulation, but with an outer `weight` instead of a `wsum`

- **Terrier**[3] [15] using two retrieval models, $PL2$ and $InL2$ with the default parameters.
- **Zettair**[4] using the default Okapi BM25 ranking algorithm [10].
- **Ancestry.com**[5] The ranked boolean system used on the main search site of Ancestry.com.

The output of these systems is a message ranking, which must be converted to a thread ranking for our assessment and evaluation. We apply three different aggregation methods to convert each message-ranking to a thread ranking for each system, resulting in a total of 24 different thread rankings in our pool. These aggregation methods are listed below:

- **Mean**: Thread score is the mean score of the retrieved messages, interpreted as a document mixture model with uniform document weights when the underlying post retrieval algorithm is a query likelihood model.
- **Max**: Thread score is the max score of the retrieved messages.
- **Pseudo-Cluster Selection** (PCS) [18]: Thread score is the geometric mean of the top-k retrieved messages. In this work, $k = 5$.

These aggregation methods were selected as representative of both *inclusive* and *selective* aggregation methods, as well as the methods found superior in preliminary studies of thread retrieval in other online forums [8, 18].

For each system, we retrieved 1000 messages per query, then retained the top 100 threads after converting to a thread ranking. Taking a union of these thread rankings across retrieval runs results in 374 unique threads per query on average.

3.2 Pooled Document Selection

Give the retrieval runs described above, we then must select those documents most likely to be relevant for judgement. Simple methods for document selection are typically employed at TREC, such as selecting all the top-k documents

[3] Terrier version 3.0, `http://terrier.org/`

[4] Zettair Version 0.9.3, `http://www.seg.rmit.edu.au/zettair/`

[5] Provided by Ancestry.com based on the simplified keyword queries, not the original structured queries.

from each run to judge [20]. This results in a maximum of $n \times k$ documents per query to judge, where n is the number of systems submitting to the pool. At TREC, frequently k is set to 100.

In our case, we do not have the resources to judge a great number of documents, and we prefer to perform a more selective method for choosing documents for the pool. We opt to apply a simple meta-search algorithm, Borda count [1], to prioritize documents retrieved by the systems. This algorithm assigns "votes" to each document based on the rank at which it was retrieved by each system. Borda count assigns a single score to each thread, with documents receiving higher scores if more systems tend to rank them highly.

Given these pooled document scores, we can then prioritize our document selection while assessing in order to favor those more likely to be relevant. Section 4 describes the algorithm for selecting document pairs for assessment and how this score is used during that selection.

4 Assessment Process[6]

In this section, we describe the process for collecting relevance assessment. Ancestry.com provided personnel to perform the assessment of forum threads. These assessors, although familiar with genealogical research, did not have experience in relevance assessment. Many of the decision made during the relevance assessment process were in an effort to simplify and streamline the assessment task for these assessors who did not have prior experience as information analysts or relevance assessors.

Several approaches to document assessment for retrieval evaluation have been proposed. Absolute judgements of a document's relevant with regard to a query have been collected at venues such as TREC [20]. Recently, the collection of document-pair preferences has been proposed as an alternative assessment methodology [5]. Initial research into the collection pairwise preferences shows that collecting the pairwise approach offers several advantages: (1) there is no need to assign an arbitrary ordinal scale to judgement levels, (2) preference capture more document ordering information than coarse absolute judgement levels, (3) assessors can assign preferences faster than absolute judgments, (4) agreement across assessors on preferences is as high or higher than on absolute judgements. The higher levels of agreement and faster assessment times are also an indication that the preference assessment task may be easier for assessors previously unfamiliar with annotating relevance. Additionally, traditional retrieval evaluation measures such as Mean Average Precision and Precision at cutoffs adapted to pairwise preferences correlate extremely highly with the traditional counterparts [3]. For these reasons, we chose to collect document pair preferences rather than absolute judgements.

Our approach to preference collection is similar to that described by Carterette, et al [5], presenting side-by-side document pairs (L, R) and collecting the assessments:

- Document L is preferred to document R ($L > R$)
- Document R s preferred to document L ($L < R$)
- Document L and document R are duplicates ($L = R$)
- Document L is *bad* (i.e. completely non-relevant)
- Document R is *bad* (i.e. completely non-relevant).

These preferences are essentially a binary judgement of which document is better. We allow assessors to identify *bad* documents and duplicates in order to avoid needlessly collecting redundant preferences or preferences on documents unrelated to the query.

In order to ensure comparability of our collection to ones previously built with pair preferences, our assessment tool is modeled after the one presented by Carterette, et al [4]. See Appendix A for a screenshot of the interface. The assessors who performed the assessment task were provided by Ancestry.com and were trained to use the assessment interface and given the assessment guidelines shown in Appendix B.

[6]The code for the assessment system and document pair selection algorithm is available to download here: `https://github.com/jelsas/django-assessment`.

In addition to the document-pair preferences we collect, we can also use these preferences to infer binary relevance judgements. If a document is ever preferred to any other document for a query, we assume the document is at least minimally relevant. If a document is judged *bad*, we assume the document is non-relevant. Duplicate documents retain the judgement of the other document in the pair, and no assumption is made about documents that are presented to the assessor but never preferred. These assumptions are consistent with the assessor instructions. We will refer to these as *inferred binary judgements* in the discussion below.

4.1 Document Pair Selection Algorithm

The naïve approach to preference assessment would require a quadratic increase in the number of assessments to collect in order to achieve a complete labeling of the documents. Carterette, et al [5] describe several methods for reducing the number of preferences necessary to collect: (1) collecting *bad* judgements and dropping those documents from further assessment, (2) assuming preference transitivity, and (3) ordering preferences to collect based on their utility in distinguishing rankings given an evaluation measure and stopping collection early. The first two methods result in sub-$O(n \ln n)$ judgements per query for a complete labeling. The third method results in an approximately linear number of judgments per query for a complete labeling, but increased complexity in the selection algorithm. We integrate the first two of these suggestions into our pair selection algorithm, but not the third.

We develop an algorithm for active selection of document pairs for assessment adhering to the above principles and several others listed below. These are based in part on learnings from previous work on annotating pairwise preferences [4, 2].

1. After each preference is collected, the preferred document of the pair should be kept in a fixed location in the interface if possible. Fixing a document's location and indicating this to the assessor eliminates the need for the assessor to re-read the document after each judgement.

2. The assessor should be given the opportunity to view the best document for the query as soon as possible, either by exposing the assessor to the entire collection or by showing "better" documents first.

3. Avoid showing the same document to an assessor too many times to avoid assessor fatigue and collect preferences over more diverse documents.

The method developed for active pair selection is presented in Algorithm 1, 2, 3 and 4. Throughout these algorithms, we assume the list of documents D is maintained in some fixed order, with documents more likely to be preferred occurring earlier in the list. We initially order D by the Borda count [1] of the document from the retrieval pool. To avoid overly biasing towards the original retrievals, we then divide D into bins of size 5 and randomize within the bins.

Algorithm 1, AnnotateDocumentPairs(), presents the general framework for active selection of document pairs. In this procedure, we build a list of preference judgements (L, R, p) with L and R documents presented on the left and right of the interface, and p a preference value, described above. This algorithm eliminates documents from the assessment pool that have been presented more then a fixed number of times k or have been marked bad.

Algorithm 2, AssessedWith(), calculates the set of documents that already have a preference assigned with a given document. The algorithm is sensitive to a parameter indicating whether we are assuming preference transitivity. If assuming transitivity, the preference graph and reverse preference graph are constructed and Djikstra's algorithm is run to identify all other documents reachable from the current document. Due to the use if Djikstra's algorithm, this procedure runs in $O(n^2)$ each time it is called, where n is the number of documents seen so far. It may be advantageous in some cases to pre-compute all paths via the Floyd-Warhsall algorithm [6] ($O(n^3)$) after each preference collection, but we do not investigate that modification here. If not assuming transitivity, the algorithm runs in $O(|P|)$ time.

Algorithm 3, FreshPair(), selects a new unseen document pair for assessment when neither the L or R documents should be held constant from the previous assessment.

Algorithm 4, NextPair(), attempts to select a new document pair when either the L or R documents are held fixed from the previous assessment. If no new pair can be identified, this procedure falls-back to calling the FreshPair() procedure (Algorithm 3).

Algorithm 1 AnnotateDocumentPairs(): General document selection strategy for pairwise preference assessment.

Input: List of documents D of length at least 2, target number of preferences to collect m, the maximum number of appearances of a document k (default ∞)
Output: List of document pairs and preferences: $\{(L, R, p)|L, R$ documents, p preference$\}$

```
P ← []
while |P| < m do
    if P empty then
        (L, R) ← FreshPair(D, P)
        Collect preference p on pair (L, R)
        Push (L, R, p) onto P
    else
        D' ← {d|d ∈ D, d has been seen ≤ k times and d has not been marked bad}
        (L, R) ← NextPair(D', P)
        if (L, R) = nil then
            return P
        end if
        Collect preference p on pair (L, R)
        Push (L, R, p) onto P
    end if
end while
return P
```

Algorithm 2 AssessedWith(): Algorithm to calculate which other documents have been judged with the given document. Note: this algorithm handles the assumption of transitive or intransitive preferences.

Input: A document d, and previously collected preferences P
Output: A set of documents that have been judged with d.

```
if Collecting transitive preferences then
    Graph G = {d_i → d_j | d_i preferred to d_j or duplicates}
    Graph G' = {d_j → d_i | d_i → d_j ∈ G}
    Find all reachable nodes in G and G' from node d (eg. via Djikstra's algorithm [6])
    return The resulting set of reachable nodes.
else
    return {d' | (d, d', p) or (d', d, p) ∈ P for any p}
end if
```

Algorithm 3 FreshPair(): Algorithm to select a fresh pair of documents from the given list of documents and previously assessed preferences.

Input: List of documents D of length at least 2, previously collected preferences P
Output: A pair of documents (L, R) or *nil* if no pairs are available.

```
for all L ∈ D do
    D' ← D \ AssessedWith(L, P)
    if D' not empty then
        return (L, D'[0])
    end if
end for
return nil
```

Algorithm 4 NextPair(): Algorithm to select the next pair of documents for assessment.

Input: Document list D, non-empty previously collected preference list P
Output: Pair of documents (L, R) for assessment, or *nil* if no more documents pairs are available.

```
(L, R, p) ← peek at last item in P
if p = L preferred or R bad or L and R duplicates then
    if L has been seen ≥ k times then
        return FreshPair(D, P)
    end if
    D' ← D\ AssessedWith(L, P)
    if D' empty then
        return FreshPair(D, P)
    else
        return (L, D'[0])
    end if
else // In this case, p = R preferred or L bad
    if R has been seen ≥ k times then
        return FreshPair(D, P)
    end if
    D' ← D\ AssessedWith(R, P)
    if D' empty then
        return FreshPair(D, P)
    else
        return (D'[0], R)
    end if
end if
```

5 Pilot Assessment & Assessor Analysis

In order to validate our approach to preference assessment, and to test whether previous findings on preference assessment hold for our assessors and test collection, we conduct a pilot assessment of 50 queries each annotated by two assessors. We do not assume transitivity in the document pair selection algorithm for this study in order to test the hypothesis that judgements are transitive. We collect up to $m = 100$ document pairs for assessment in this pilot. We also do not place a limit on the number of times a document can be shown to the assessor, letting $k = \infty$ in Algorithm 1 above.

We evaluate three questions with the pilot assessment:

1. What level of *external* agreement for pairwise preferences exists between assessors? I.e. if one assessor expresses a preference, does the other assessor agree?
2. What level of *external* agreement for inferred binary relevance exists between assessors?
3. What level of *internal* agreement exists among a single assessors preferences? I.e. does an assessor produce transitive preferences?

The agreement shown in Tables 5, 6 and 7 aim to answer these questions.

In Table 5, we show for each pair of documents (A, B), the preference label assigned by the two assessors. Only explicit preferences are considered in this table, and no transitivity is assumed. We do make the assumption that documents that have ever been preferred for that query are preferred to any marked *bad*. The overall agreement between the two assessors (excluding the "no pref" row and column) is 22.6%. This, however, counts equally all judgement decisions, whether they are for an expressed preference or an assessor deemed the two documents not relevant for the query. One assessor, corresponding to the column counts, is clearly much more aggressive in identifying *bad* documents. If the two assessors have differing thresholds for declaring a document *bad*, as we see here, a preference between documents stated by one assessor may appear as two bad documents to another assessor.

		Assessor 1				
		$A < B$	$A > B$	A, B bad	$A = B$	no pref
Assessor 0	$A < B$	381	126	1078	2	856
	$A > B$	143	304	986	10	0
	A, B bad	0	0	1	0	0
	$A = B$	2	0	1	0	1
	no pref	185	241	1744	2	342

Table 5: Preference inter-assessor agreement for all document pairs A, B summed across all 50 queries used in the pilot assessment. Note: Due to active selection strategy, not all pairs shown to both assessors and a document pair may be presented in opposite order.

We can look at this problem of pairwise agreement between assessors from an analytical perspective and calculate a bound on the maximum preference agreement two assessors will have if one is more aggressive in in marking *bad* documents. Assume, for example, both assessors perfectly agree on the total orderings of n documents. One assessor has a more conservative view of the information need and marks some fraction s of documents *bad*, $0 \leq s \leq 1.0$, while the other assessor marks no documents as bad. If $s = 0$, then agreement is 100%, but if $s > 0$, pairwise agreement will suffer even though both assessors agree on the total ordering of our documents. The overall agreement between these two assessors is given by:

$$\text{Agreement} = \frac{\binom{n}{2} - \binom{ns}{2}}{\binom{n}{2}} = \frac{n(n-1) - ns(ns-1)}{n(n-1)} = 1 - \frac{ns^2}{n-1} + \frac{s}{n-1}$$

where we have $\binom{n}{2}$ total document pairs and $\binom{n \times s}{2}$ document paris marked both *bad* by one assessor. As $n \to \infty$ the agreement between our assessors approaches $1 - s^2$ (and is quite close to $1 - s^2$ for values of $n \approx 40$ as in our dataset). In our case, one assessor assigns effectively no documents a *bad* label, and the other marks *bad* roughly 64% of the document seen (see Table 6). This places an upper bound on their agreement at 59%, assuming they perfectly agree on the absolute ordering of all the document seen. The observed agreement of 22.6% is roughly 39% of this upper bound.

Although this discrepancy exists between our assessors' willingness to assign a *bad* label, it primarily indicates the assessors had differing interpretations of the specificity of the information need. Our primary concern is agreement between assessors when they both express a definite preference, shown in the shaded cells in the table. When only considering these cells of the table the agreement is much higher at 71.8%. This is an acceptable level of agreement for preferences, and comparable to previous levels of agreement observed for preference assessment [5]. The discrepancy in assessors' willingness to assign a *bad* judgement was addressed in further assessor training and clarification of the assessment guidelines.

Table 6 shows agreement on inferred binary judgements. Based on the differences observed above in how the

		Assessor 1	
		relevant	non-relevant
Assessor 0	relevant	333	591
	non-relevant	2	3

Table 6: Inter-assessor agreement on inferred binary judgements, counts summed across queries. Those documents that have ever been preferred are considered relevant, those documents marked *bad* are considered non-relevant. No assumption is made on documents that are presented but never preferred.

assessors label *bad* documents, we expect to see a lower level of agreement when looking at inferred binary relevance. This is the case here, where we see a 36.2% overlap in relevance judgements (size of the intersection of relevant documents divided by the union). This level of agreement is at the lower end of the range of agreement levels for absolute judgements observed in other studies using more experienced assessors [5, 19], and well below the level of agreement we observed on preferences.

Finally, we must validate whether the assessors produce internally consistent judgements – that is, whether the assessors' preference judgements are transitive. To calculate the fraction of an assessor's judgements that are transitive, we compute for each (assessor, query) pair a statistic t, the fraction of document triples that the assessor judged transitively:

$$P = \{(A, B, C) | \text{ Assessor prefers } A > B \text{ and } B > C\}$$
$$T = \{(A, B, C) | (A, B, C) \in P \text{ and Assessor preferrers } A > C\}$$
$$t = \frac{|T|}{|P|}$$

The t values, macro-averaged across queries, are shown in Table 7. Additionally, we show the number of queries queries that each assessor assessed perfectly transitive, i.e. with $t = 1.0$. In this table, we can see that the assessors' judgements are almost perfectly transitive. Again, this finding agrees with previous work [5] and indicates that we can assume preferences are transitive when collecting assessment on more queries.

	Macro-Average t	**Queries Perfectly Transitive**
Assessor 0	0.991	24
Assessor 1	0.999	48

Table 7: Internal consistency of assessors' judgements over 50 queries. t measures the fraction of document triples assessed transitively.

6 Additional Assessment

Based on learnings from the pilot study, we conduct an additional assessment of 141 other queries, each assessed by one assessor. We adjust the parameters of the pair selection algorithm based on the following findings of the pilot study:

- Preference annotations are almost perfectly transitive.
- Annotating 100 document pairs per query leads to assessor fatigue.
- If a highly preferred document was presented early in the assessment, this document tended to be presented as one item of the pair in all the remaining pairs.

Based on these findings, we assume transitivity in the document pair selection algorithm, and collect up to $m = 60$ document pairs to be assessed per query. We also limit the number of times a single document can be presented, setting $k = 5$ in the selection algorithm. Table 8 shows the assessment statistics for the *name* and *name+* queries.

Query Set	Num. Docs Seen	Pairs Assessed	Num. Bad	Num. Duplicate
name	44.10	69.16	9.68	0.29
name+	46.09	69.79	20.89	0.24

Table 8: Ancestry.com assessment statistics per query set. Statistics shown averaged across queries in each set.

7 Pairwise Preference Evaluation[7]

The collection of pairwise preferences necessitates evaluation measures computed over those preferences, rather than traditional absolute judgements. Previous work has proposed several analogues to absolute evaluation measures computed over preferences [3]. We use most of these measures in our evaluation, with some modifications to $APpref$, described below.

In all these measures, we assume the set of preferences, P, is given:

$$P = \{(A, B)|A, B \text{ documents s.t. } A \text{ is preferred to } B\}. \tag{1}$$

This set of preferences is either explicitly given by the assessors, or implicitly through an assumption of transitivity, for example. Each document in the set of preferences has an associated rank assigned by the ranking algorithm, $\rho(A)$. We can then define the set of *correctly ranked* preferences:

$$P_{\text{corr}} = \{(A, B)|(A, B) \in P \text{ and } \rho(A) < \rho(B)\} \tag{2}$$

where the ranking of the documents corresponds with the preferred ordering.

The analogue to Precision at a cutoff ($P@k$) is *ppref@k*, the fraction of correctly ordered preferences where at least one documents in the pair is ranked above k. Formally, this is given by:

$$ppref@k = \frac{|\{(A, B)|(A, B) \in P_{\text{corr}} \text{ and } \min(\rho(A), \rho(B)) \leq k\}|}{|\{(A, B)|(A, B) \in P \text{ and } \min(\rho(A), \rho(B)) \leq k\}|}. \tag{3}$$

Similarly, an analogue to Recall at cutoffs ($R@k$), *rpref@k*, is defined as the fraction of *all* preferences that meet the criteria above. Formally, this is given by:

$$rpref@k = \frac{|\{(A, B)|(A, B) \in P_{\text{corr}} \text{ and } \min(\rho(A), \rho(B)) \leq k\}|}{|P|}. \tag{4}$$

An analogue to Average Precision (AP) over preferences, *APpref*, has also been proposed. Carterette, et al define this as "ppref [...] averaged over ranks at which rpref increases" [3]. This definition, however, is somewhat problematic. First, *ppref* and *rpref* values change at different ranks, so some *ppref* values may not get factored into the calculation. Second, because some *ppref* values may be ignored, it is possible to construct a ranking with some incorrectly ordered pairs but achieves a "perfect" *APpref* of 1.0. Third, the original definition does not state how to treat unranked pairs, whereas typically Average Precision calculations factor in a minimum precision value for all unranked relevant documents [12].

We propose an alternative formulation of *APpref* that captures all *ppref* values regardless of whether *rpref* changes at that rank level. First, we define the set of *preferred* documents P^+ as those that ever been preferred to any other document:

$$P^+ = \{A|(A, B) \in P\}. \tag{5}$$

Note that the ranks of documents in P^+ are the only ranks where *rpref* can change, although it does not necessarily change at all those ranks. We then average $ppref$ values over the ranks of all preferred documents to calculate $APpref$:

$$APpref = \frac{\sum_{A \in P^+} ppref@\rho(A)}{|P^+|} \tag{6}$$

If we define $\rho(A) = \infty$ for those documents unranked by the retrieval system, excluding those pairs from the set P_{corr}, this definition of $APpref$ behaves similarly to Average Precision for unranked documents.

[7]The code for evaluation of retrieval system output with pairwise preferences is available for download here: `https://github.com/jelsas/Pairwise-Preference-Evaluation`

8 Pool Results

Table 9 gives the performance of each system and aggregation method used in the document pool. We show both $ppref@10$ and $mAPpref$.

Note that not all the message ranking algorithms used in the pool are interpretable as a query likelihood probability, $P(Q|M)$, as in Indri. For those other message retrieval models, we apply similar mathematical transformations to the message scores to produce a thread score.

System	Aggregation Method	$mAPpref$	$ppref@10$
Indri BOW	$MEAN$	0.5428	0.4925
	MAX	0.5998*	0.5615*
	PCS	0.6569+*	0.6402+*
Indri DM	$MEAN$	0.5492	0.5067
	MAX	0.6089	0.5716
	PCS	**0.6612**+*	**0.6466**+*
Indri fielded `#wsum`	$MEAN$	0.4628	0.4291
	MAX	0.5296	0.4985
	PCS	0.5663	0.5420
Indri fielded `#weight`	$MEAN$	0.5422	0.4938
	MAX	0.6105	0.5693
	PCS	0.6572	0.6105
Terrier $PL2$	$MEAN$	0.5286	0.4934
	MAX	0.5820	0.5381
	PCS	0.6475	0.6225
Terrier $InL2$	$MEAN$	0.5406	0.4837
	MAX	0.5828	0.5297
	PCS	0.6554	0.6230
Zettair $BM25$	$MEAN$	0.5001	0.4586
	MAX	0.5654	0.5271
	PCS	0.6300	0.6083
Ancestry.com	$MEAN$	0.4357	0.4036
	MAX	0.4702	0.4481
	PCS	0.5380	0.5808

Table 9: Pool system performance results for Ancestry.com. Significance tested for shaded rows only. Significant gain at the $p < 0.01$ level over BOW $MEAN$ and MAX models indicated with * and + respectively.

There are two dimensions along which we can evaluate these results — the aggregation method, and the underlying retrieval system. First, looking at the aggregation methods, the results here agree with those on the MacRumors.com dataset. As before, we see a clear superiority of the selective models (MAX, PCS) over the inclusive model ($MEAN$). For every input message ranking algorithm, and both $ppref@10$ and $mAPpref$, we observe that the $MEAN$ model is inferior to the MAX model, which is inferior to the PCS model. We test significance of the three message aggregation methods using the Indri Bag of Words ranking model, and find that both $PCS > MAX$ and $MAX > MEAN$ at the $p < 0.01$ level.

Next, looking at the underlying message retrieval systems, these results are generally well aligned with our expectations. None of the systems used in the pool use any training data to set parameters, relying on default values or intuition to set field weights and smoothing levels. Indri and Terrier are modern, well maintained and widely used retrieval systems that consistently do well at the Text REtrieval Conference (TREC) (see, for example, in the Blog

Track [11, 16, 17]). The Terrier models do not use term proximity or order, and the results with the $InL2$ model are almost indistinguishable from the analogous Indri bag-of-words (BOW) model. Indri's Dependence Model (DM) does leverage term proximity and order, and as expected performs slightly better than the models without this information. The performance difference between Indri's BOW and DM models is somewhat less than observed in other collections [13] and none of these differences are significant for this task and collection.

Interestingly, the fielded retrieval models do not perform as well for this task as expected. Typically, using within-document field structure in retrieval models yields significant performance improvement in tasks such as known page finding and xml element retrieval [14]. In the models used for these experiments the fields and field weights were chosen in a somewhat ad-hoc manner, using intuition and a handful of sample queries to choose the weights before any relevance assessment was performed. For this reason, it is unlikely those weights are optimal for this task, and further tuning of weights using training data will likely result in improved performance. Investigating the use of fielded retrieval models for forum search will require further experiments, but is out of the scope of the current work.

The Zettair search engine uses an implementation of the Okapi BM25 retrieval model [10] and is designed to be compact and fast search engine. This search engine is also used actively at TREC, performing generally well [9, 21]. It is unclear why Indri and Terrier consistently outperform Zettair in the thread search task, possibly having to do with Zettair's default parameters not being well suited to this tasks.

The Ancestry.com retrieval system uses a ranked boolean model, a less sophisticated keyword ranking model than the other systems evaluated. This retrieval model is designed for use with richly structured query forms, including name, location and date fields as well as keywords. For the purposes of these experiments, all fields are treated as keywords, possibly contributing to its lower performance as compared to other systems.

9 Conclusion

This report outlines the creation of the Ancestry.com Online Forum Test Collection. We present an analysis of the document collection, including observations of the usage volume over time and general distributions of various structural elements in the collection. We also present an analysis of a query set provided by Ancestry.com and show the primary types of information needs present in the users of their search tool.

As a supplement to the document collection we built an information retrieval test collection for studying thread search. To do this, we simulated a pool of retrieval systems, sampled documents to assess and collected preference assessment over document pairs. Finally, we showed the performance of the different systems used to create the document pool.

There are many other language technology, natural language processing and social network analysis tasks that may be appropriate to study with the Ancestry.com Online Forum document collection. We discuss a thread search task here, but other tasks may include:

- **Sub-forum Search**: Any node in the hierarchical organization of the online forum site may be a reasonable result to return for users seeking out a appropriate place to ask a specific question.
- **Author Search**: Many forum sites provide the ability to send private messages to individuals. A novice user many want to use this facility to directly ask an expert a question, rather than post a open question to the entire forum user base. In this case, providing facilities for author, or expert search in online forums may be particularly useful.
- **Response Prediction**: Given the start message of a thread, can the number of responses, speed of response, or authors who respond be predicted automatically?
- **Authority Measures**: Some authors of messages in the collection may have more expertise than others. Are there measures of author expertise that can be developed to help in document retrieval performance?

References

[1] Javed A. Aslam and Mark Montague. Models for metasearch. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 276–284, New York, NY, USA, 2001. ACM Press.

[2] Paul N. Bennett. Personal communication, September 2010.

[3] Ben Carterette and Paul N. Bennett. Evaluation measures for preference judgments. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 685–686, New York, NY, USA, 2008. ACM.

[4] Ben Carterette, Paul N. Bennett, and Olivier Chapelle. A test collection of preference judgments. In *SIGIR 2008 Workshop: Beyond Binary Relevance: Preferences, Diversity and Set-Level Judgment*, SIGIR'08, 2008.

[5] Ben Carterette, Paul N. Bennett, David Maxwell Chickering, and Susan T. Dumais. Here or there: preference judgments for relevance. In *Proceedings of the IR research, 30th European conference on Advances in information retrieval*, ECIR'08, pages 16–27, Berlin, Heidelberg, 2008. Springer-Verlag.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[7] Nick Craswell. W3C test collection, April 2005.

[8] Jonathan L. Elsas and Jaime G. Carbonell. It pays to be picky: an evaluation of thread retrieval in online forums. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 714–715, New York, NY, USA, 2009. ACM.

[9] Steven Garcia. Rmit university at trec 2009: Web track. In *Proceedings of the Eighteenth Text Retrieval Conf*, TREC '09, 2009.

[10] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.*, 36:779–808, November 2000.

[11] Craig Macdonald, Iadh Ounis, and Ian Soboroff. Overview of the TREC 2007 blog track. In *Proceedings of the Sixteenth Text Retrieval Conference*, TREC '07, 2007.

[12] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[13] Donald Metzler and Bruce W. Croft. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479, New York, NY, USA, 2005. ACM Press.

[14] Paul Ogilvie. *Retrieval using Document Structure and Annotations*. PhD thesis, Carnegie Mellon University, 2010.

[15] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: A High Performance and Scalable Information Retrieval Platform. In *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*, 2006.

[16] Iadh Ounis, Craig Macdonald, Maarten de Rijke, and Gilad Mishne. Overview of the TREC 2006 blog track. In *Proceedings of the Fifteenth Text Retrieval Conference*, TREC '06, 2006.

[17] Iadh Ounis, Craig Macdonald, and Ian Soboroff. Overview of the TREC 2008 blog track. In *Proceedings of the Seventeenth Text Retrieval Conference*, TREC '08, 2008.

[18] Jangwon Seo, W. Bruce Croft, and David A. Smith. Online community search using conversational structures. *Information Retrieval*, (to appear) 2011.

[19] Ellen M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 315–323, New York, NY, USA, 1998. ACM.

[20] Ellen M. Voorhees. The philosophy of information retrieval evaluation. In *CLEF '01: Revised Papers from the Second Workshop of the Cross-Language Evaluation Forum on Evaluation of Cross-Language Information Retrieval Systems*, pages 355–370, London, UK, 2002. Springer-Verlag.

[21] Mingfang Wu, Falk Scholer, and Steven Garcia. Rmit university at trec 2008: Enterprise track. In *Proceedings of the Seventeenth Text Retrieval Conf*, TREC '08, 2008.

[22] Wensi Xi, Jesper Lind, and Eric Brill. Learning effective ranking functions for newsgroup search. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 394–401, New York, NY, USA, 2004. ACM.

A Ancestry.com Assessment Interface

Figure 6 shows the assessment interface for collecting preference annotation.

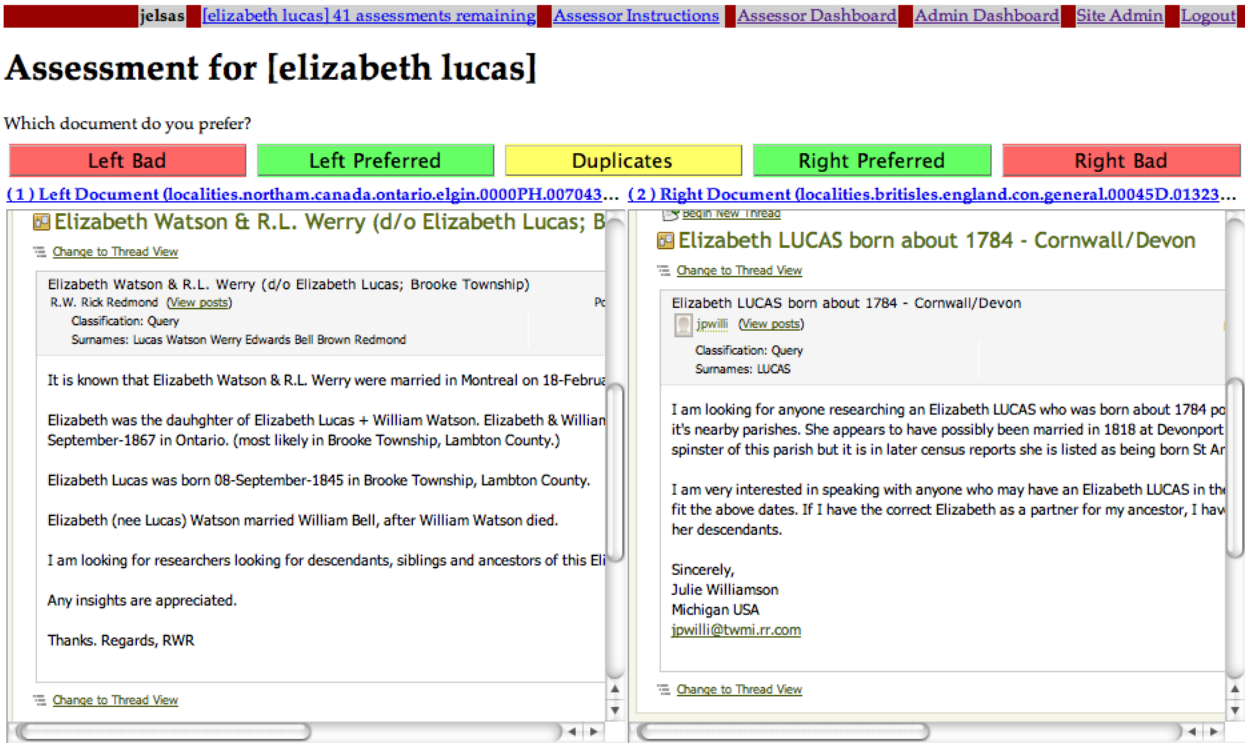

Figure 6: Ancestry.com assessment interface

B Ancestry.com Assessment Guidelines

The following assessment guidelines were used as instructions for the Ancestry.com data annotation:

> The user issuing the query is seeking information on the person, people or family named in the query. Some queries contain more detailed information, such as locations, dates, occupations or a race. Your task is to compare two search results for the query, and select the one that is most useful.
>
> Please follow the guidelines below when making decisions on the utility of a document. These guidelines are meant to give some reasons for favoring one document over another, but there may be other reasons one document could be preferred. This is inherently a subjective task, and in some cases there will not be a clear best answer.
>
> - Many queries are ambiguous and could refer to several different people with the same name. Only mark a document Bad if it cannot plausibly refer to the person in the query.
> - If you see an error loading a document, please click the link above the document containing the text "Left Document..." or "Right Document..." to re-load the document. If that does not work, then mark the document Bad.
> - If the two documents are duplicates, eg. have nearly the same text and the same author, then mark them as Duplicates.
> - Documents primarily discussing the person are the most useful, for example an obituary, or detailed description of the persons family.
> - Documents mentioning the person in passing, for example referencing the person as a sibling or spouse, are less useful.
> - Documents containing incidental mentions of the person, for example in a passenger listing, are less useful.
> - Documents with no mention of the person are not useful and should be marked Bad.
> - Documents providing information should be preferred over those that are only requesting information.
> - Documents providing more complete or thorough information should be preferred over those providing less information.
> - Documents by authors who may be an expert in genealogical research or an expert on this person or family should be preferred over those by authors who appear to be novices or are uninformed. For example, some authors may say they have previously researched a person or family, or may say they are new to genealogical research.