# Ranking Algorithms For Digital Forensic String Search Hits

*By*

**Nicole Beebe and Lishu Liu**

*Presented At*

The Digital Forensic Research Conference

**DFRWS 2014 USA** Denver, CO (Aug 3rd - 6th)

# Ranking Algorithms for Digital Forensic String Search Hits

Nicole L. Beebe, Ph.D.
The University of Texas at San Antonio
DFRWS2014

# Acknowledgement/Disclaimer

This publication was developed under work supported by the Naval Postgraduate School Assistance Agreement No. N00244-11-1-0011, awarded by the Naval Supply Systems Command (NAVSUP) Fleet Logistics Center San Diego (NAVSUP FLC San Diego). It has not been formally reviewed by NPS. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the NPS or NAVSUP FLC San Diego. The NPS and NAVSUP FLC San Diego do not endorse any products or commercial services mentioned in this publication.

# Overview

- Background
  - Information overload problem with string search results
  - Why ranking algorithms are possible solution
- Research
  - Identification of relevancy ranking features
  - Ranking algorithm development
    - Machine learning model building and ranking functions
  - Empirical results
    - Relevant/non-relevant (class) prediction accuracies
    - Relevancy ranked list (score) precision, recall, average precision
    - Feature significance analysis
- Conclusions, software, next steps

# BACKGROUND

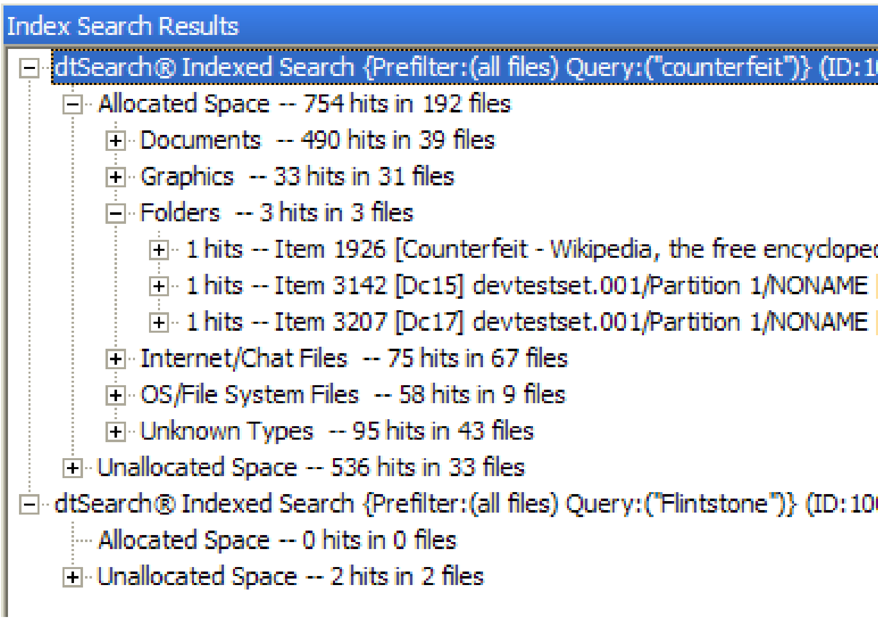
# Motivation

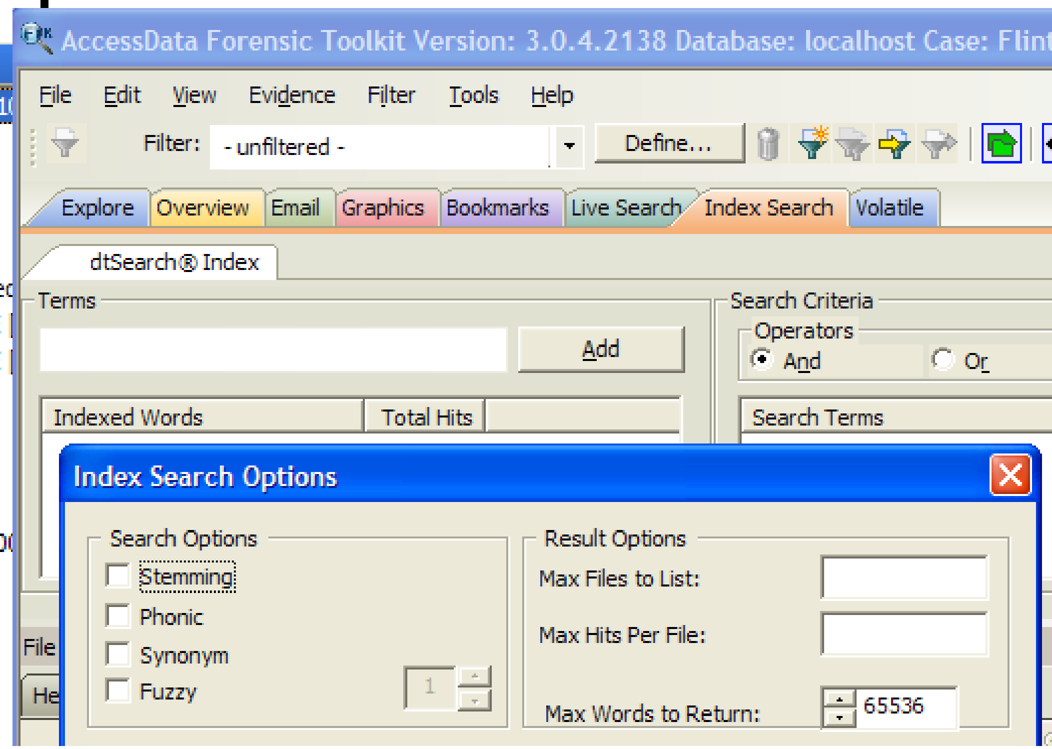
- String searching nearly infeasible, yet still worthwhile
  - Much info/evidence sought is textual in nature
  - Extremely low signal to noise ratio (<5%)
  - Millions+ hits for reasonably small queries
  - Resource constraints favor other search techniques

- Current attempts to solve the problem
  - State of the art DF tool features *adding* to noise
  - Cluster-based platforms for increased compute power
  - Hit sorting (query, data type, allocation status)
  - Some improvement via grouping by object type

# DIGITAL FORENSIC STRING SEARCH OUTPUT

## What We Have…



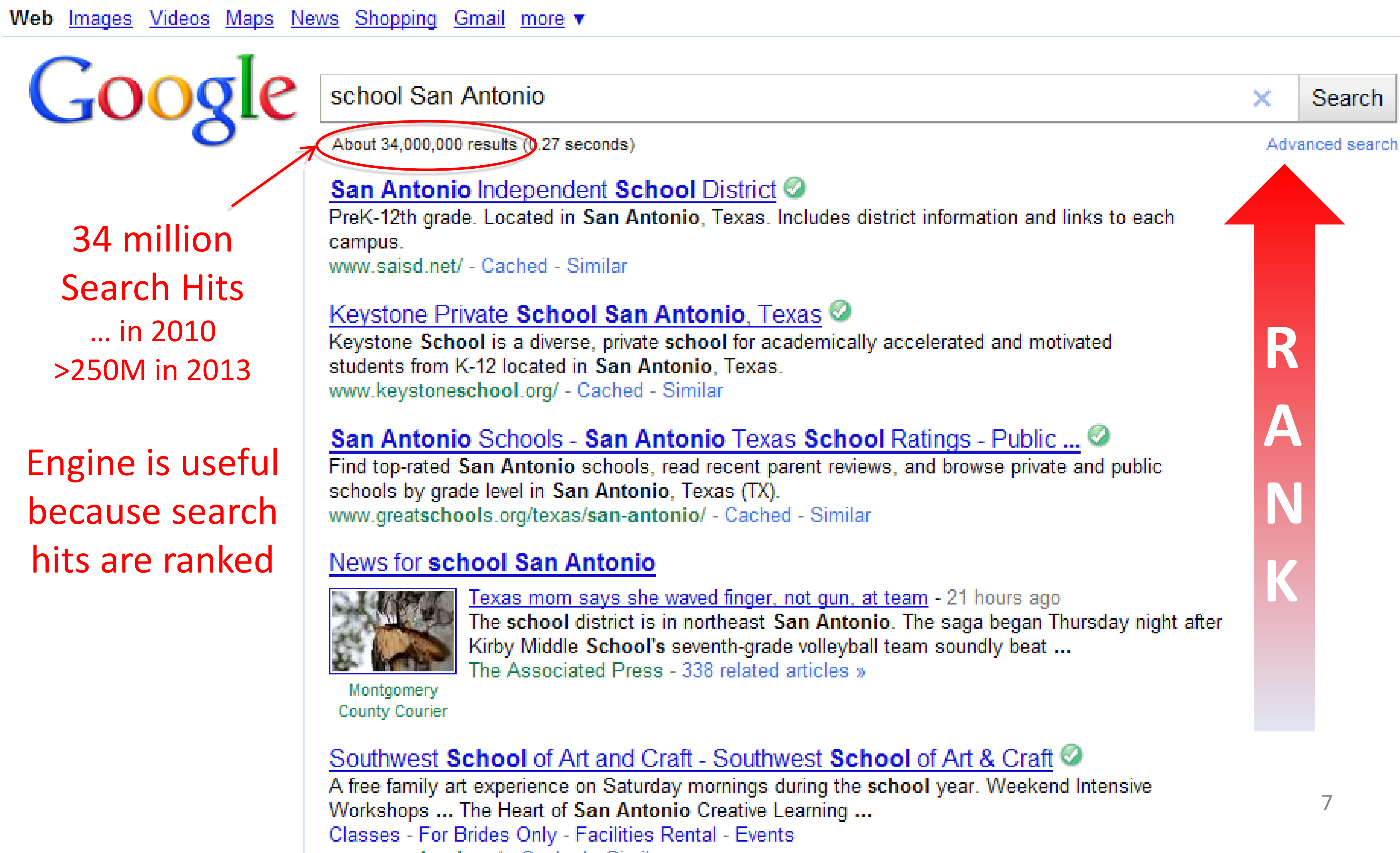




- Hit grouping

  Query based, Data type, File type/item

# DIGITAL FORENSIC STRING SEARCH OUTPUT

## What We Want…



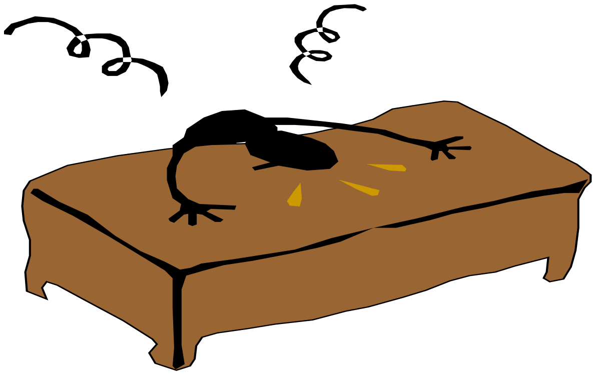

34 million Search Hits … in 2010
>250M in 2013

Engine is useful because search hits are ranked

# In short…

What would "Googling" be like
without ranking algorithms?

… Ask a digital forensic analyst!



# Problem is only getting worse…

+

=

# Search Hit Ranking

Simulated Digital Forensic Text String Search Hit Output:

| Search Hit | Rank Score |
|---|---|
| I plan to kill her after dark tonight… | 3.5 |
| …kill killed killer killing… | 1.4 |
| kill -9 3303 | 0.8 |

So… just “Google” it.

If it were only that simple…

- We need to identify appropriate ranking features for <u>this</u> domain.
- Few of Google’s 200+ features apply in the digital forensics context.

# THE RESEARCH

# Research Overview / Methodology

1. Theorized 18 quantifiable characteristics (AKA ranking features)
2. Trained a support vector machine (SVM) to generate ranking functions
   – Binary class SVM feature weights can be used in a weighted, linear ranking function
3. Empirically tested ranking functions
   – Achieved 81.02%-85.97% prediction accuracies
   – Significant improvement in average precision over unranked lists (0.82 & 0.90 vs. 0.50*)

*artificially high, due to balanced data set—equal number of relevant & non-relevant hits

# STEP 1: Feature Identification

- Theorized quantifiable characteristics (ranking features)
  - of allocated files and unallocated clusters containing hits
  - of the string search hits themselves
  - believed pertinent to hit relevancy determination
  - based on past ranking research, existing ranking applications, and investigator experience

# Ranking Feature Specifics

| Feature | Description | Operationalization |
|---|---|---|
| Recency-Created | Temporal proximity of an allocated file's creation to a reference point | Data extracted from the $STANDARD_INFORMATION attribute from $MFT records; difference between date/time stamp and a reference point (specified as date of forensic analysis in this case, but may differ in other cases); normalized by maximum time difference in corpus (difference between oldest date/time stamp and reference point); continuous feature with range f={0…1}, with lower values being closer to reference point |
| Recency-Modified | Temporal proximity of an allocated file's modification to a reference point | |
| Recency-Accessed | Temporal proximity of an allocated file's access to a reference point | |
| Recency-Average | Average MAC temporal proximity to a reference point | |
| Filename-Direct | Hit exists in a file/path name | Simple pattern match operation for the hit's search expression in the file's path/filename; binary feature with f={0\|1} |
| Filename-Indirect | Hit is contained in the content of an allocated file, whose file/path name contains a different search term. | Simple pattern match operation for other search expressions in the file's path/filename; binary feature with f={0\|1} |
| User Directory | Hit is contained in an allocated file found in a non-system directory | Specified standard Windows system directories and defined user directories as all non-system directories; binary feature with f={0\|1} |

Note: These features are only applicable to hits found in allocated space; Driving the need for separate allocated vs. unallocated ranking functions.

# Ranking Feature Specifics

| Feature | Description | Operationalization |
|---|---|---|
| High Priority Data Type | Hit is contained in a high priority data type | Specified high-medium-low data type tables; used file signatures of allocated files for type identification; used Sceadan, a naïve statistical data type classifier, for data type classification of unallocated blocks; binary feature with f={0\|1} for each priority level |
| Medium Priority Data Type | Hit is contained in a medium priority data type | |
| Low Priority Data Type | Hit is contained in a low priority data type | |
| Search Term TF-IDF | Term frequency moderated by inverse document frequency of the search term in the corpus | Used normalized, logarithmic, corpus level term frequency, moderated by inverse document frequency (see Eq. 2); continuous feature with range f={0…1} |
| Block-level hit frequency | Count of instances of the search hit term in an allocated file or cluster | Measured by the term frequency (TF) of the search expression in the file or unallocated cluster; normalized by the highest TF returned; continuous feature with range f={0…1} |
| Cosine-Similarity | Traditional cosine similarity between query and file/cluster vector | Measured by the traditional IR cosine similarity measure between the document and the query; normalized by the highest cosine similarity measure returned; continuous feature with range f={0…1} |

# Ranking Feature Specifics

| Feature | Description | Operationalization |
|---|---|---|
| Search Hit Adjacency | Byte-level logical offset between adjacent hits (next nearest neighbor) | Distance (in bytes) between search expression and the most proximally located search hit for a different search expression; measured via file offset to account for fragmentation effects on distance; normalized the largest adjacency distance returned; continuous feature with range f={0…1} |
| Search Term Block Offset | Distance from start of file or unallocated cluster | Measured by file offset of the search expression from the start of the file or cluster; normalized by largest search term block offset value returned; continuous feature with range f={0…1} |
| Proportion of Search Terms in Block | How many different search terms appear in the file or cluster | Total number of search expressions that exist in the file or cluster; normalized by the maximum number of search expressions per block returned; continuous feature with range f={0…1} |
| Search Term Length | Byte length of search term | Search expression's length in bytes; normalized by maximum length of any search expressions; continuous feature with range f={0…1} |
| Search Term Priority | User ranked priority of search term | Measured by rank-ordering of the search expressions by the user; normalized by the highest numeric rank returned; continuous feature with range f={0…1} |

# STEP 2: Ranking Function Development

- Trained a binary class (relevant/non-relevant), linear kernel, support vector machine (SVM)
  - Generate SVM model with feature weights
  - Use binary class feature weights as coefficients in ranking functions (fast linear discriminant functions)

$$R_{hit} = \sum_{n=1}^{18} w_n f_n$$

  - Traditional SVM would assign threshold for class prediction
  - Linear discriminant function approach facilitates continuous scale relevancy rank *score*

# Data Set & Sampling

- M57 Patents case ("police seizure images")
  - http://digitalcorpora.org
  - 4 user workstations imaged on last day of scenario
- Executed 36-term search query
  - 2.6M search hits in 46.9K files/clusters
  - 4.24% relevancy rate (determined by human analyst*)
- Search hit sample selection
  - All relevant hits
  - Random sample of non-relevant hits to create balanced sample (equal number of relevant and non-relevant)



*Some inferences made; see paper

# Model Building

- Used `libsvm` and `liblinear`
- Experimentally selected linear kernel
  - Experimentally selected optimal solver, parameter values
- Used 60%:40% train:test ratio during model building & testing (random sampling without replacement)
- Trained two classifiers – allocated & unallocated
  - Since not all features are applicable to unallocated

# STEP 3: Empirical Testing

Allocated Model Confusion Matrix

| True / Predict | Not Relevant | Relevant |
|---|---|---|
| Not Relevant | 75.2% | 24.8% (false pos.) |
| Relevant | 13.2% (false neg.) | 86.8% |

Unallocated Model Confusion Matrix

| True / Predict | Not Relevant | Relevant |
|---|---|---|
| Not Relevant | 63.3% | 16.7% (false pos.) |
| Relevant | 5.8% (false neg.) | 74.2% |

- False positive rate exceeded false negative rate
  - Preferred in this context, to avoid missing relevant evidence
  - Could fine-tune the relevancy ranking threshold if desired

# But…What about <u>relevancy score</u> performance?

- Less interested in binary class prediction
  - relevant vs. non-relevant determination
- More interested in relevancy ranking score for ranked list ordering of string search hit output:

| Search Hit | Rank Score |
|---|---|
| I plan to kill her after dark tonight… | 3.5 |
| …kill killed killer killing… | 1.4 |
| kill -9 3303 | 0.8 |

# Relevancy Score, Ranked List Performance

- Calculated relevancy rank score ($R_{hit}$) for hits
- Created relevancy rank ordered search hits list
- Measured average precision
- Measured precision & recall at quartile increments

$$\text{Average Precision (AvgP)} = \frac{\sum_{r=1}^{N} P(r) \times \text{rel}(r)}{R}$$

where $r$ = rank

$N$ = number hits retrieved
rel($r$) = 0 or 1 (relevancy of hit)
$P(r)$ = total precision up to this point
$R$ = Total number of relevant hits

# Ranked List Performance

**Allocated Model**

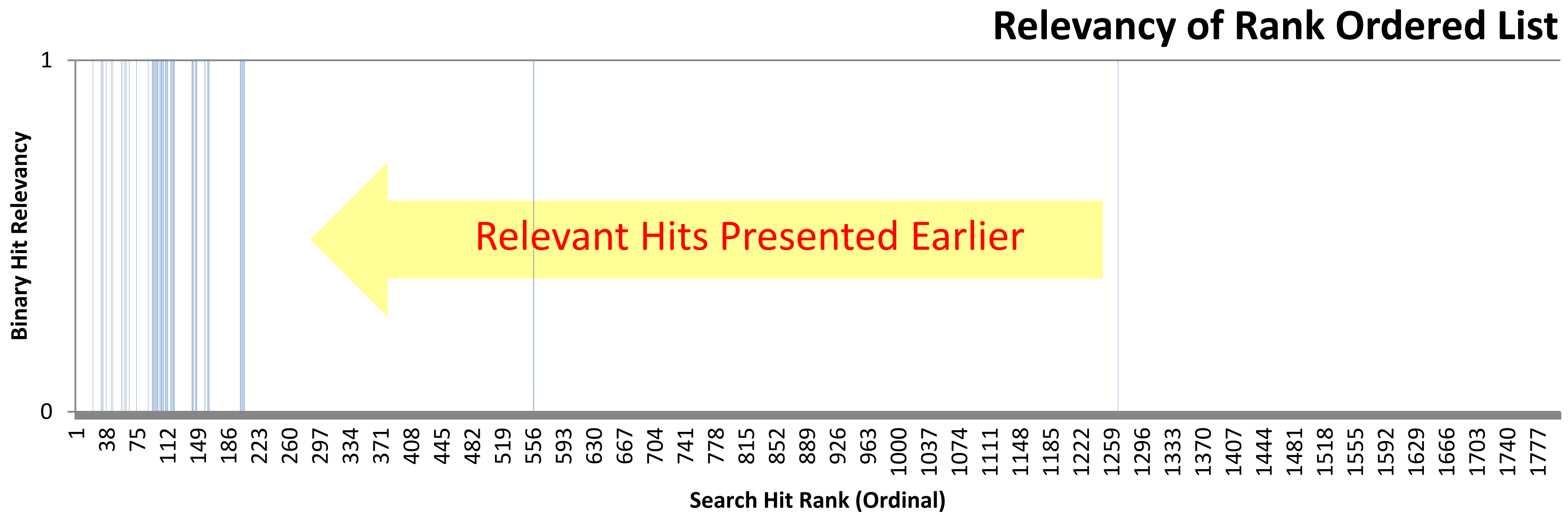
| No. Hits Retrieved | Recall | Precision | Average Precision |
| --- | --- | --- | --- |
| 25% | 0.42 | 0.84 | 0.37 |
| 50% | 0.80 | 0.80 | 0.68 |
| 75% | 0.96 | 0.64 | 0.80 |
| 100% | 1.00 | 0.50 | 0.82 |

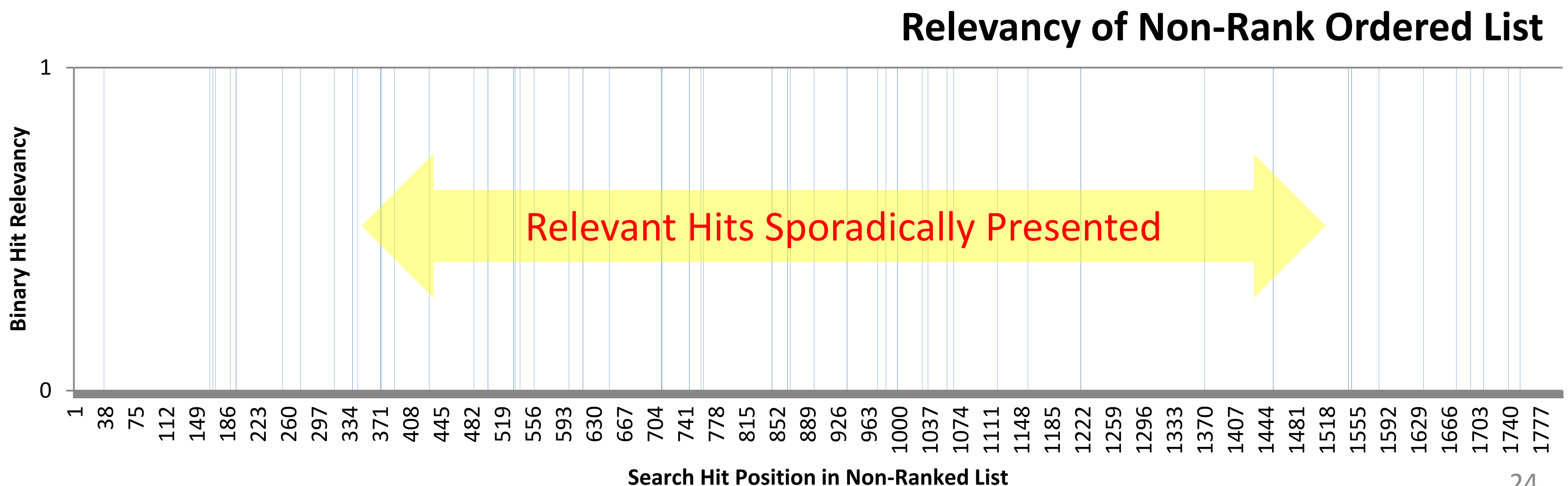
**Unallocated Model**

| No. Hits Retrieved | Recall | Precision | Average Precision |
| --- | --- | --- | --- |
| 25% | 0.46 | 0.92 | 0.43 |
| 50% | 0.86 | 0.86 | 0.79 |
| 75% | 1.00 | 0.66 | 0.90 |
| 100% | 1.00 | 0.50 | 0.90 |

- Conclusion: Helps analyst find relevant hits faster!

# Visualization of Ranked List Performance*






*different case; for visualization only

# Which Features Seem to Matter Most?

- Relative absolute magnitude of feature weight is a measure of feature significance
- Most significant features in both models
  - Search term length
  - Search term priority
  - TF-IDF of search term
  - Proportion of search terms in an object
- Most significant features in the allocated model
  - Filename features
  - User vs. system directory
  - Some date/time stamp features
  - Search term object offset
- Most significant features in the unallocated model
  - Object-level hit frequency

# Which Features Seem to Matter Least?

- Some date/time stamp features
- Data type prioritization
- Cosine similarity
- Search hit adjacency

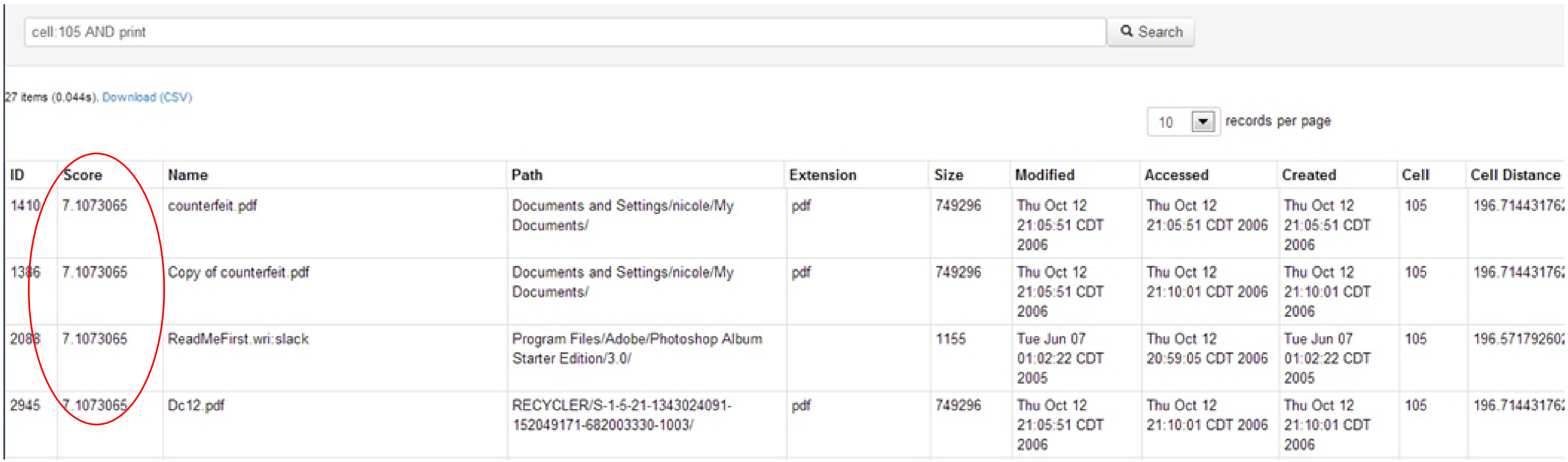
# SUMMING IT UP…

# Conclusions & Limitations

- Search hit ranking algorithms are feasible
- Search hit ranking algorithms are fast
  - No performance results reported (sorry)
  - Slows down evidence processing slightly, but not much
- Search hit ranking algorithms can save significant analyst time spent wading through non-relevant hits
- Limitations
  - Single, synthetic case
  - Need real-world data to better train/test ranking functions

# Current Capability & Next Steps

- Ranking algorithms are currently implemented in open source tool (`Sifter`)



cell:105 AND print

Search

27 items (0.044s). Download (CSV)

10 records per page

| ID | Score | Name | Path | Extension | Size | Modified | Accessed | Created | Cell | Cell Distance |
|---|---|---|---|---|---|---|---|---|---|---|
| 1410 | 7.1073065 | counterfeit.pdf | Documents and Settings/nicole/My Documents/ | pdf | 749296 | Thu Oct 12 21:05:51 CDT 2006 | Thu Oct 12 21:05:51 CDT 2006 | Thu Oct 12 21:05:51 CDT 2006 | 105 | 196.71443176 |
| 1386 | 7.1073065 | Copy of counterfeit.pdf | Documents and Settings/nicole/My Documents/ | pdf | 749296 | Thu Oct 12 21:05:51 CDT 2006 | Thu Oct 12 21:10:01 CDT 2006 | Thu Oct 12 21:10:01 CDT 2006 | 105 | 196.71443176 |
| 2088 | 7.1073065 | ReadMeFirst.wri:slack | Program Files/Adobe/Photoshop Album Starter Edition/3.0/ | | 1155 | Tue Jun 07 01:02:22 CDT 2005 | Thu Oct 12 20:59:05 CDT 2006 | Tue Jun 07 01:02:22 CDT 2005 | 105 | 196.57179260 |
| 2945 | 7.1073065 | Dc12.pdf | RECYCLER/S-1-5-21-1343024091-152049171-682003330-1003/ | pdf | 749296 | Thu Oct 12 21:05:51 CDT 2006 | Thu Oct 12 21:10:01 CDT 2006 | Thu Oct 12 21:10:01 CDT 2006 | 105 | 196.71443176 |

- Currently modifying `Sifter` to collect real-world training data from beta-test volunteers/users
- Plan to validate/improve generic models and create additional case type specific models

Nicole.Beebe@utsa.edu

210-458-8040 (w) 210-269-5647 (c)

# COMMENTS?? QUESTIONS??