RJ10364 (A0510-011) October 28, 2005
Computer Science

IBM Research Report

Disambiguation of References to Individuals

Levon Lloyd
Department of Computer Science
State University of New York
Stony Brook, NY 11794-4400

Varun Bhagwan, Daniel Gruhl
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

Andrew Tomkins
Yahoo Research
701 First Avenue
Sunnyvale, CA 94089

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Disambiguation of References to Individuals

Levon Lloyd
Department of Computer Science
State Univeristy of New York
Stony Brook, NY 11794-4400.
lloyd@cs.sunysb.edu

Varun Bhagwan
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120.
vbhagwan@us.ibm.com

Daniel Gruhl
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120.
dgruhl@us.ibm.com

Andrew Tomkins
Yahoo Research
701 First Ave
Sunnyvale, CA 94089.
atomkins@yahoo-inc.com

ABSTRACT

We study the problem of disambiguating references to named people in web data. Each name spotted online is shared by several hundred people on average, and teasing apart these references is critical for a new family of person-aware analytical applications. We present and evaluate algorithms for this problem, and give results to indicate that 25% of personal references may be successfully disambiguated with precision in excess of 95%, but that larger fractions cause a significant decline in precision.

General Terms

Algorithms, Experimentation, Measurements

Keywords

Name Disambiguation, Clustering

1. INTRODUCTION

In the beginning, the web acted primarily as a large-scale distributed hyperlinked document store, with browsing and search as the access methodologies. Now, however, users are applying the web in their work and lives for a much broader range of functions, and new targeted applications are beginning to emerge to support this shift. Social network providers allow users to treat their social network as an asset. Competitive intelligence companies maintain dossiers on the online presence of certain corporations and key figures. Compliance applications track references to suspicious characters. In many of these new application areas, a solid sense of personal identity is a necessary precursor: nobody would enjoy being turned down for a date or a bank account because another individual with the same name was known to engage in illegal or immoral activities. Yet these types of scenarios are becoming increasingly plausible.

Furthermore, the problem of name ambiguity is widespread and will only grow worse as a greater fraction of the population arrives online. The US Census Bureau reports the most common first and last names among its survey pool. Assuming independence of first and last names, the average over all people in the US Census of the number of people in the census with that name is 450.[1] Thus, when a random candidate walks into a bank office or barroom, there will be on average 450 other people in the US with the same first/last name. When we extend to the global population, the situation only worsens.

In this paper, we study the problem of disambiguating textual references to individuals.

Consider the hypothetical Professor John Smith, a teacher and researcher in polymer chemistry. Six years ago, some of his publications and professional appearances appeared online. Today, however, there is information online sufficient to uncover that he runs triathlons, coaches his daughter's soccer team, and is a member of his local school board.

Of course, a naive search for John Smith would also reveal that he teaches computer science, poetry, urban studies, is a welder, a doctor, a butcher, a baker, coaches swim teams, baseball teams and cricket teams, plays 32 different sports, arranges flowers professionally, and has over a dozen felonies to his name.

Disambiguating between all the possible John Smiths is important not only for identifying information about an individual, but also for understanding the networks of business, professional and social relationships to which the Web provides access.

Among humans, we recognize the possibility of confusion, and when authoring we include linguistic clues to help avoid conflating instances. For example, we might refer to "Prof. Smith *of Springfield City College*." Likewise, as readers we look for clues in the document to determine whether it might refer to the correct John Smith. An article about a soccer game John Smith is coaching in Anchorage, Alaska probably refers to a different person.

Our goal in this work is to develop algorithms capable of

[1]Without the assumption of independence, this number will be larger, as two people with the same first name are more likely to share a last name due to cultural affinities between certain names.

clustering references to a particular name so that the resulting clusters correspond as closely as possible to the particular individuals bearing that name. We explore the use of linguistically derived features and bottom up clustering to explore how well this disambiguation can be performed.

1.1 Applications of Identity Determination

Applications employing names are numerous and growing, and include the following:

Dossier Creation Create a "dossier" containing information about a given person from publicly available sources.

Relationship Detection Discover other people, corporations, countries, organizations, etc., that are related to this person.

Person Search Search for information regarding a person. The information sought might be a homepage, media mentions, place of employment, or other information.

Expertise Location Help me discover a person who has the skills and the willingness to help me solve some problem. Tools for expertise location are more common in the enterprise than on the general web.

Authorship Find postings, articles, or books authored by this person.

Homepage Location Discover the homepage for a given person. This application has come into recent prominence due to its incorporation in the NIST TREC series of information retrieval benchmarks

1.2 The case for high precision

Name disambiguation is naturally cast as a clustering problem in which references to a given name are the points to be clustered, and the true underlying clusters are the distinct individuals who share that name. This formulation is appropriate, but care must be taken in defining the appropriate measure of correctness. A number of metrics for evaluating such clusterings have been defined [11, 13, 2], which are either explicitly based on, or track closely, an analogue of F-measure. In this analogue, the precision is the probability over all references that co-clustered references are the same person, and the recall is the probability over all references that an instance of the same person is co-clustered. F-measure is then computed using the standard definition.

In our problem, however, we wish to support automated inferencing with a very small probability of failure to avoid the situation in which an individual is denied resources due to conflation with a completely different person who happens to share a name. Thus, maintaining a sufficiently high precision is paramount, and recall may then be improved as much as possible. We define a framework for clustering in this type of environment.

There is an obvious class of applications where the inverse is true – when you cannot afford to miss anything. The trivial implementation here is to read every page; clearly there are other approaches that allow tremendous increase in precision for very small decrease from perfect recall, but these are not explored in this paper.

2. RELATED WORK

We were introduced to the problem of name disambiguation by work of Garg and Guha [5], who address disambiguation of names across the web as we do, but with a focus on a user seeking references to a particular individual via a search interface.

The natural language community has focused on name disambiguation as a class of word sense disambiguation. Work of Wacholder et al.[18] on the "Nominator" system focused on both cross-document and same-document identification of name co-references. Mann and Yarowsky [10] combined natural language approaches with template-based meta-data as background information. Smith and Crane [17], and later Amitay et al. [1], study disambiguation of place names rather than people names.

On the web, however, the problems of name disambiguation and word sense disambiguation are quite different. The number of different possible senses of the word is known a priori and is typically very small(< 20), while in name disambiguation the number of instances of a name is quite large (over 400) on average, and certain names ("John Smith," for example) on the web may have many thousands of referents. Tools like dictionaries, thesauri, and foreign language translations can all be used to solve the word sense disambiguation problem; however, the space of people's names is unbounded in size and suitable reference works do not exist.

Bagga and Baldwin[2] propose using a vector space model to disambiguate names. They assign a vector, corresponding to the words in the contexts of the occurrences, to each ambiguous document and say that two documents refer to the same person if the distance between their vectors is below a threshold. They also introduce the B-cubed algorithm, which we use here, for evaluating the output of a disambiguation system.

Gooi and Allen[6] give a thorough study of name disambiguation techniques on a large scale corpus. They compare different clustering techniques and metrics over the vector space model used in [2]. They conclude that agglomerative clustering gives the best and most robust results, but on larger sets of data, incremental clustering was much more efficient.

One family of approaches to name and other entity disambiguation is to apply templated-based extraction mechanisms to find meta-data about the entity that can be used in disambiguation. Riloff [16] proposed an iterative extraction technique that discovered rules and then instances, and used the instances to uncover more rule templates. Jones et al.[9] formalized bootstrapping as a partial alternative to expensive training mechanisms, allowing automated application of template extraction techniques. Brin [3] proposed a similar scheme for extracting relational information from pages on the web.

There are also a number of commercial applications, such at the name analysis software of LAS, Inc., and the name resolution software of SRD, that focus on matching and understanding of references to individual as their core competency.

The `citeseer` project focuses on name resolution in particular with reference to authors in citations [8]. Pasula et al.[14] study this problem from a generic machine learning perspective of identity uncertain. Novak et al.[13] consider the opposite problem: rather than asking how different people with the same name may be separated, they consider

Arnold Schwarzenegger	Cameron Diaz	Kevin Bacon
Kevin Spacey	Nicole Kidman	Harrison Ford
	Sharon Stone	

Table 1: Famous Actors

Nicholas Negroponte	Allen Newell	Jon Kleinberg
George Cybenko	Larry Stockmeyer	John Hennessy
Gilbert Strang	Andrew Wiles	Brian Kernighan

Table 2: Notable Computer Scientists/Mathematicians

how the same person operating under different names (or aliases) may be brought together. This dual but related problem requires similar measurements and techniques.

3. DATA

In this section, we describe our process for generating a set of names to experiment upon, and then our data gathering and cleaning procedures.

3.1 Sets of names

We created two distinct data sets, one based upon household names that have a significant web presence, and another based on "standard" names of people who are referenced reasonably often on the web. Our well-known individuals are drawn from two classes: famous actors and actresses who are known to the general population, and famous computer scientists and mathematicians whose names typically occur with high frequency online even if they are not familiar to the general populace. Table 1 lists the actors and actresses we use, and Table 2 lists the computer scientists and mathematicians. These sets are taken from earlier work on co-occurrence graphs based on web mentions [4].

Our second set of one thousand names is generated from analysis of web data. We scanned a set of approximately seventy million distinct names mined from a subset of just over two billion web pages in IBM's WebFountain project [7]. From these names, we restricted to those which occurred at least five hundred times within the 2.1B web pages we examined for name spotting. We then used information the 1990 US Census to estimate the probability that a uniform person in that census would match both the first and last name, assuming that first and last names are chosen independently. We required that this probability be below 5×10^{-8}. From this set, we chose one thousand names at random for our experiment.

3.2 Data gathering and cleaning

We used the full 2.8B pages of the IBM's WebFountain system [7] to gather data and run experiments regarding references to names within our sets. For each name, we performed an index query to generate references to that name. For each result, we extracted a region of one hundred words centered around the name, and replaced each occurrence of the first and last name within that region with `FIRST` and `LAST` respectively; thus, "Nicole Kidman's performance in Cold Mountain..." becomes "FIRST LAST's performance in Cold Mountain...". A randomly-selected set of snippets from this data set is shown in Table 3. As the table shows, the snippets are quite noisy, very sparse in features, and present a significant challenge to clustering algorithms. Note, the examples in Table 3 are from the actors and cs/math datasets. The "1000 random" set of names generated even lower quality snippets.

Pennsylvania in 1973, with a dissertation focusing on the computational modeling of human problem solving performance in the tradition of FIRST LAST and Herbert Simon. George Luger had a five year postdoctoral research appointment at the Department of Artificial Intelligence of
Maid In Manhattan Movie Show Times Coming To Video Stores Next Tuesday Like Mike Geena Davis in Stuart Little 2 FIRST LAST in Trapped Harrison Ford in K-19 The Widowmaker Web Sites K-19 The Widowmaker home subscribe .pick list . tell
EST SACRAMENTO, Calif. - The Rincon Band of Mission Indians filed a suit in federal court on June10 accusing Gov. FIRST LAST of unfair compact negotiations with other tribes. The suit claims that by allowing tribes to lift the current cap
All Caption Image Number Headline Event Number Keyword Photographer Location Venue Bill Paxton (1) Brooke Shields (1) Emilio Estevez (1) FIRST LAST (2) Sharon Stone (2) Tom Cruise (3) Year 2004 Year 2003 Year 2002 Year 2001 Year 2000 Year 1999
FIRST LAST - Web Site The Web Site for
FIRST LAST , Jim Broadbent, John C. Reilly, Henry Thomas, Liam Neeson 2 DVD, exkluzivn vydn Strhujc epos o lidech, kte vybudovali
little trickle of news stories is distilled from the 'funnel'. Sidebar ——- 'The Daily Me' is attributed to MIT professor FIRST LAST . True personalization requires an extra step a recurring set of interactions between news provider and news consumer that permits
any two people can be linked by a string of no more than 6 mutual acquaintances, or 6 degrees of FIRST LAST , where Mr. LAST can be linked to any film star via other stars hes performed with? I just found
Quick Search Full Search L E G A L LEGAL Governor Names Gaming Negotiator January 08, 2004 Gov. FIRST LAST on Wednesday named a Bay Area attorney and former appellate-court judge to lead the state's negotiations with Indian tribes

Table 3: Example snippets from our web names dataset

While most of our feature extraction was focused on the (up to) 100-word window around the reference, we did consider extracting one clean class of feature from the entire page: references to the names of other people. This is described in more detail in Section 4.

Our experimental methodology is the following. In order to experiment with disambiguating references to k distinct individuals, we choose k distinct names from one of our sets of names and concatenate together the desired number of references to each person into a single file. This file contains only references to the meta-name "FIRST LAST." The algorithm is then asked to cluster references within this file, and the results are checked to see whether the resulting clusters match the original names.

4. ALGORITHMS

This section covers the feature extraction and clustering

algorithms we evaluate.

4.1 Feature Extraction

We examine the following features:

Keywords: *tfidf*-scored tokenized keywords from the text snippets

Entities: Any people's names occurring on the page containing the reference, as extracted by a name recognizer. Also, any entity that exists in the Stanford TAP knowledge base [15].

Descriptions: Appositives and noun phrase modifiers that modify the name reference in the snippet.

Phrases: Heads of all noun phrases in the snippet.

We now present a more detailed view of each type of feature.

Keyword Feature Extraction

We generated a set of keywords from a large set of candidate snippets using standard methodology as follows:

1. All the snippets were merged, tokenized by whitespace, and the resulting tokens were counted.
2. The top 1000 most frequent tokens were dropped, as were any terms occurring fewer than 3 times in the entire corpus.
3. The remaining set of terms were taken to be a dictionary, and the counts were saved to compute corpus frequencies.

Named entities

A very rough biographical profile of a person can be formed by looking at the entities that are associated with that person even if the nature of these associations are unknown [10]. If two people with the same name have a strong overlap in the space of associated entities, then they are probably the same person. Conversely if there is little overlap they are probably different people. We therefore extracted two families of entities for each snippet, as follows.

We processed the entire web page containing a given snippet to extract names of people on the page. The extraction was performed using a pattern-based extractor tuned for the noisy text common in web pages. This spotter has been described elsewhere [4]. We performed several experiments using only name features, and for these experiments we further processed the feature set by removing names that occurred only once or twice in the corpus, for efficiency.

We also processed the snippets themselves to extract references to entities occurring in the TAP knowledge base [15]. A sequence of tokens in the snippet matches a TAP entity if the two strings are identical, and no further disambiguation of the entity is performed. A stop list of ambiguous entities is employed to keep the ambiguity to a minimum.

Descriptions

Appositives and noun phrase modifiers are two linguistic structures that are commonly used to provide descriptions of a person. An appositive is a noun, often with modifiers, set beside another noun to explain or identify it. One has to be careful when using appositives and noun modifiers for a couple of reasons. First, two different phrases could be synonyms of one another or one could be a more specific version of another. Also, adjectives and prepositional phrases will make variations on the noun phrase that we do not want to consider.

Eagle County District Attorney Mark Hurlbert had argued...
→ Eagle County District Attorney noun phrase modifier
→ district attorney longest suffix that is a person
→ lawyer canonicalization of district attorney node

Figure 1: Description extraction from noun phrase modifiers.

Jason Williams, *point guard for the Memphis Grizzlies*,...
→ point guard for the Memphis Grizzlies appositive
→ point guard rightmost sequence of nouns not to the right of a preposition
→ guard longest suffix that is a person
→ basketball player canonicalization of guard node

Figure 2: Description extraction from appositives.

In order to extract the description from an appositive or a noun phrase modifier we do two things. First, we heuristically find the head of the noun phrase by taking the rightmost sequence of nouns that is not to the right of a preposition. Then, we canonicalize the noun using the is-a semantic hierarchy from WordNet [12] in the following manner. First we find the longest suffix of the phrase that has a sense that is a descendant of the person node in the is-a tree. Then for each sense of the phrase we canonicalize it to the lowest ancestor that has at least a threshold document frequency in the background corpus in the sub-tree that is rooted at it. This step is to make sure that we merge siblings, but do not make the descriptions overly general. This process is illustrated with noun phrase modifiers in Figure 1 and with appositives in Figure 2.

4.2 Clustering

Once the feature vectors are developed, we consider two algorithms for grouping them together: a standard K-means variant, and a bottom-up incremental clustering algorithm tailored for this problem.

4.2.1 K-Means Clustering

The K-Means used as cluster of $2 \times trueCluster$ random seed vectors in the subspace of one randomly selected input data vector. Any clusters that fell below a membership threshold (usually 5) had their centroid reseeded into the center of the largest cluster (plus a small offset). This served to break up the super-clusters that would otherwise dominate the solution, lowering recall slightly in favor of higher precision. Clustering continued until no significant improvement in convergence was seen.

4.2.2 Incremental Clustering

Motivated by the run-time performance disparity between incremental and agglomerative clustering shown in [6], our

tailored scheme is incremental in nature. We developed a custom seed generation phase since the algorithm seemed to perform much better when started on a clean set of initial clusters. We then added a cluster merging step at the end to improve recall. The parts of the algorithm are the following:

Seed generation: Generate clean but small seed clusters

Classification: Classify all references into seeds or new clusters

Merging: Merge clusters as necessary

Seed generation

The goal of the seed generation step is to form a set of highly precise seed clusters that need not cover the entire set of documents. We find such clusters using the assumption that if a feature is popular in the set of ambiguous documents, but not popular in the background corpus, then it is highly unlikely that it is associated with more than one of the ambiguous people.

We evaluate each feature in turn in *tfidf* order, and perform one of three actions:

1. If this feature has not appeared in any page that is already part of a seed cluster and occurs in more than a threshold number of pages, then we create a new seed cluster containing the pages that have this feature.
2. If this feature has appeared in another seed cluster and the ratio of the number of pages that contain both to the number that only contain this feature is greater than a threshold, then add the pages that have this feature to the seed cluster with the overlap.
3. Otherwise we skip the feature.

The motivation for this step is the following. If a new feature occurs that has very low overlap with other features, we treat it as likely that the feature belongs to a new instance of the ambiguous name. On the other hand, if a new feature overlaps significantly with an existing cluster, we assume that it is correlated with the same instance of the person, and so we merge the features into a single seed. If the feature does not either lie within an existing cluster or lie largely outside existing clusters, we skip it.

Classification

The next step of our algorithm is to classify each page that was not assigned to a seed cluster. For each page, we find the cluster that is closest to the page in the feature space. If the distance is below a threshold then we add it to that cluster. Otherwise, we find the cluster that is closest to it in our entity co-occurrence space. If that distance is below a threshold then we add it to that cluster. If the page is not close enough to any existing cluster, then we create a new singleton cluster with just this page.

Cluster Merging

Experimentation showed that the first two steps of our algorithm often created too many clusters, thus we added a final step to merge clusters. We merge clusters by repeatedly merging a cluster with its nearest neighbor in the feature space until there are no clusters that are close enough to it. We iteratively grow each cluster in this manner.

5. RESULTS

We begin this section with a detailed evaluation of k-means clustering of name references, and based on the results of these analysis, we turn to our new iterative clustering algorithm.

5.1 Evaluation of Results

Consider a set $R = \{r_1, \ldots, r_n\}$ of documents or snippets, each containing a reference to a person. Let $P = \{P_1, \ldots, P_{|P|}\}$ be a partition of R into references to the same person, so $P_i = \{r_1, r_7, r_8\}$ might for instance be a set of references to John Smith the chemistry professor. Let $A = \{A_1, \ldots, A_{|A|}\}$ be a collection of disjoint subset of R created by the algorithm. Rather than requiring that A be a partition of R, we will in some cases allow the algorithm to cluster only some of the references. We will denote by R_A the references that have been clusters by the algorithm. We write $P(r_i)$ to mean the set P_i containing reference r_i, and $A(r_i)$ to be the set A_i containing r_i, if one exists. Fix some reference $r_i \in P_j \cap A_k$ that has been assigned to some cluster by the algorithm. We define a notion of precision and recall as follows:

$$\textsc{Prec}(r_i) = \frac{|\{r \in A(r_i) : P(r) = P(r_i)\}|}{|\{r \in A(r_i)\}|}$$

$$\textsc{Recall}(r_i) = \frac{|\{r \in P(r_i) : A(r) = A(r_i)\}|}{|\{r \in P(r_i)\}|}$$

Thus, the precision of a reference to John Smith is the fraction of references in the same algorithm cluster that are also to John Smith. The recall is the fraction of references to John Smith that appear in that algorithm cluster. The average precision and recall and F-measure for the clustering A is then:

$$\textsc{Prec} = \sum_{r \in R_A} \frac{\textsc{Prec}(r)}{|R_A|}$$

$$\textsc{Recall} = \sum_{r \in R_A} \frac{\textsc{Recall}(r)}{|R_A|}$$

$$\text{F} = \frac{2 \cdot \textsc{Prec} \cdot \textsc{Recall}}{\textsc{Prec} + \textsc{Recall}}$$

These measure were introduced by Bagga and Baldwin [2] and Novak et al. [13].

5.2 Observations

We begin with a few observations. The data set represented by short snippets taken from web text is quite difficult to classify correctly—see Table 3 for several examples chosen at random from our dataset. Results are typically quite poor even for just a few different classes of names. A natural disparity between precision and recall emerges in this regime: algorithms that place several different aspects of the same person into several different clusters will be punished for their recall, while algorithms that simply create a giant cluster will attain an F-measure of 2/3 for disambiguation of two people. Thus, we cannot simply present F-measure. We will instead present all three measures, and focus as described in the introduction on techniques that generate high precision.

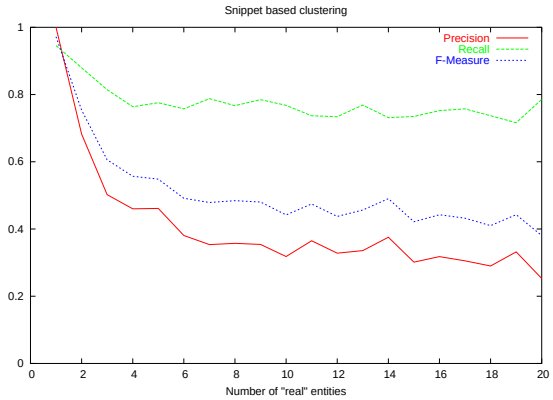

Figure 3: Plot of Precision, Recall and F-Measure for TF on snippets.

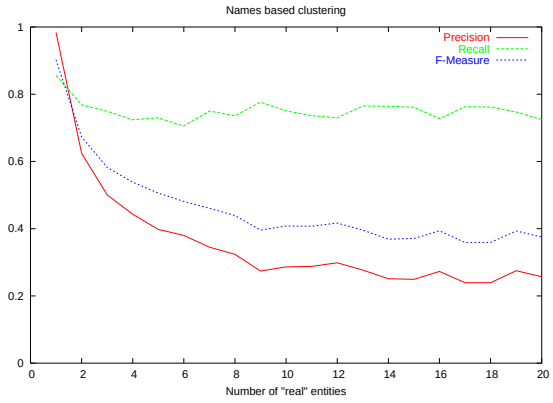

Figure 4: Plot of Precision, Recall and F-Measure for Namespotter features.

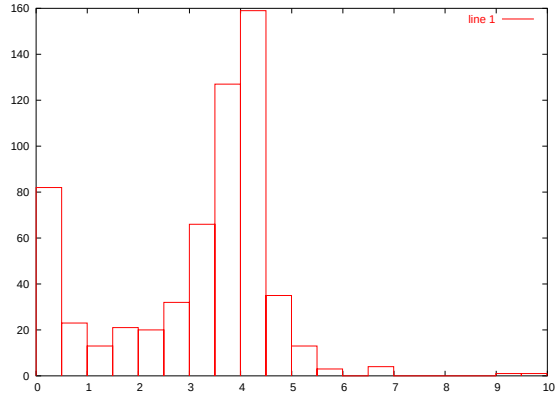

Figure 5: Histogram of cohesion scores (average distance from centroid to cluster element).

5.3 Evaluating features

Figures 3 and 4 show results for keyword features and person's name features respectively. The clustering algorithm is k-means. The number of distinct people fed to the algorithm varies from two to twenty. The high recall suggests that large clusters are being maintained by the algorithm; however the precision at twenty names is 0.25, five times higher than creation of a single cluster containing all names would allow. In fact, the quality decays only slowly after the input reaches 6 or 8 people, suggesting that some fraction of the references to each additional person are easily disambiguated. The same pattern holds out to 50 users.

The experiments performed with only names as features performs surprisingly well: results are quite similar for all numbers of entities. The names feature set is quite sparse, but as the results show, snippets that do not contain a name are also unlikely to contain high-quality features that can be used for successful clustering.

5.4 Focus on precision

The results of the previous section provide low precision, but suggest that certain of the clusters may in fact be more clean. We first verified that many smaller high-precision clusters were being found by the algorithm, and then considered the following algorithmic and experimental modification.

We compute the *cohesion* of a cluster as the average distance from the cluster centroid to the elements of the cluster. Study of the histogram of cohesion values in Figure 5 shows that many clusters display almost the same cohesion as a set of random vectors, while a few clusters display much smaller values. In order to focus on precision, we give the algorithm the ability to *endorse* certain clusters as appearing to be of high quality. We then ask for the algorithm to endorse an increasing fraction of the overall data, and consider the trend in precision and recall as this fraction increases. Figure 6 shows the results. At approximately 10% of the data, the algorithm is able to select clusters of near-perfect precision.

High-precision clusters are interesting only to the extent that they contain a significant number of data points. We consider the case of 10 distinct names with 200 snippets per name, resulting in 2000 data points to be clusters. Figure 7 shows the mean number of data points per cluster for each

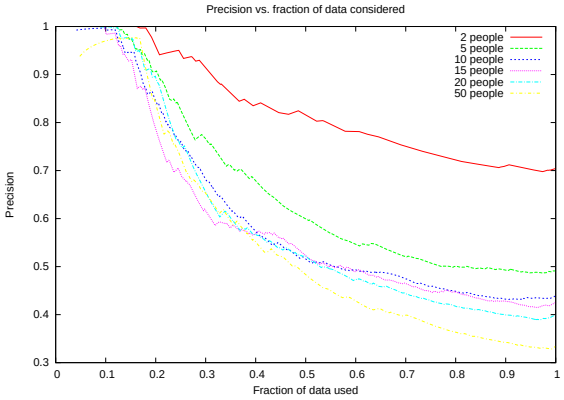

Figure 6: ROC style curves for cohesion check. Note we are using faction of data rather than recall here.

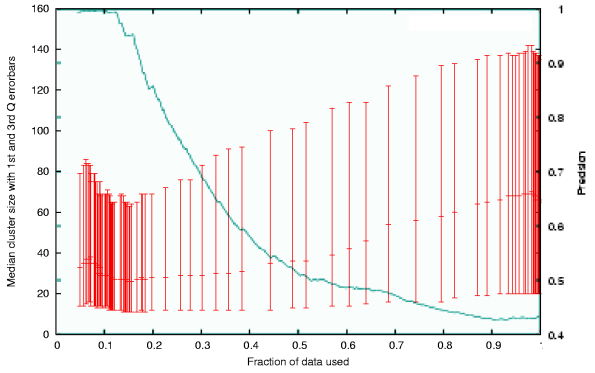

Figure 7: Average cluster size with first and third quartiles for 10 person case as a function of the fraction of data endorsed by the algorithm. Overlaid curve is precision of the clusters at that fraction of data. Left y axis labels show size of cluster; right labels show precision.

fraction of the data endorsed by the algorithm. Matching this plot to the precision values of Figure 6 shows that at 15% of the data being endorsed, the precision is in excess of 95%, and the mean cluster size is about 27 people. The errorbars on the figure show the cluster size for the first and third quartiles of endorsed data points; these values are computed by sorting the k endorsed data points by the size of the cluster containing the point, and then returning the cluster sizes corresponding to the data points at index $k/4$ and $3k/4$. At 95% precision and 15% endorsed data, the middle 50% of the endorsed data belong to high-precision clusters containing 15–70 references. With 200 references per person, each of these clusters captures between 7% and 35% of the overall references to that person.

5.5 Incremental clustering

We now consider the incremental clustering algorithm defined in Section 4.2. Recall that the incremental algorithm consists of three phases: seed generation, classification, and

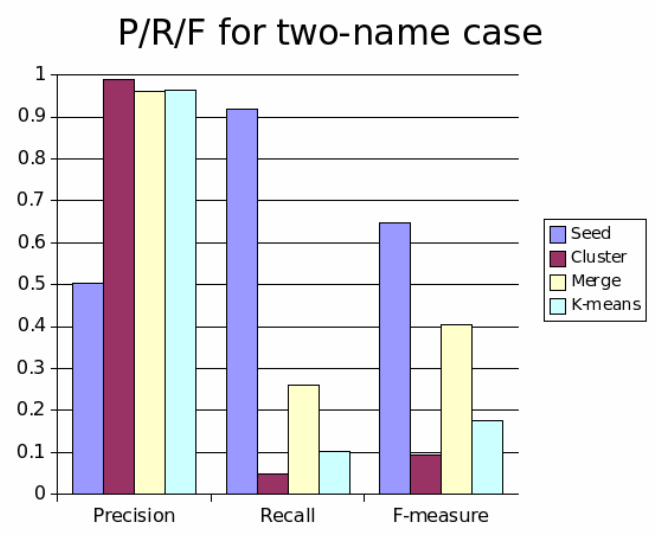

Figure 8: Comparison of precision, recall and f-measure, for our algorithm at the 3 steps (seed, cluster and merge) as well as k-means for reference.

merging. We consider terminating the algorithm after each phase and considering the results, as follows. After seed generation, we may terminate the algorithm and output all unclassified data points as a single large cluster. After classification, we may terminate the algorithm and output the cluster as it stands. And after merging, we simply allow the algorithm to output its final clusters. Figure 8 compares the algorithm at all three points to the k-means algorithm discussed above.

After seed generation, the algorithm has produced high-quality seed clusters, but the remaining documents have all been clustered together, resulting in high recall and low precision. Once the classification step completes, the precision approaches 1, but recall has dropped dramatically since the classifier is wary of placing questionable material into an existing cluster. The merging phase then brings clusters together to provide a recall of approximately 25% with a tiny drop in precision down to about 0.96. The k-means algorithm produces clusters with comparable precision but recall around 0.1. Thus, the incremental clustering algorithm achieves 250% of the recall at a similar precision.

6. CONCLUSION

We have presented a technique for disambiguating occurrences of an ambiguous name from snippets of web text referring to individuals. To enable applicability to domains where the cost of a false positive is high, we have targeted the high-precision end of the ROC curve. In our technique, the disambiguating algorithm is allowed to declare a certain fraction of mentions too difficult to disambiguate, but it must return high precision results for the remainder of the data. We then evaluate the algorithm based on the fraction of the data it chooses to score, and its recall within that fraction. We show that, over typically web references, it is possible, with linguistically enhanced feature vectors and an incremental classifier, to return results for 25% of the data with precision in excess of 0.95, out-performing the non-enhanced approach by a factor of around $\times 2.5$.

7. REFERENCES

[1] E. Amitay, N. Har'El, R. Sivan, and A. Soffer. Web-a-where: geotagging web content. In *Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 273–280. ACM Press, 2004.

[2] A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th international conference on Computational linguistics*, pages 79–85. Association for Computational Linguistics, 1998.

[3] S. Brin. Extracting patterns and relations from the world wide web. In *Selected papers from the International Workshop on The World Wide Web and Databases*, pages 172–183. Springer-Verlag, 1999.

[4] C. Faloutsos, K. McCurley, and A. Tomkins. Fast discovery of *Connection Subgraphs*. In *Proceedings of the Tenth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2004.

[5] A. Garg and R. Guha. Disambiguating entities in web search.

[6] C. H. Gooi and J. Allan. Cross-document coreference on a large scale corpus. In *HLT-NAACL*, 2004.

[7] D. Gruhl, L. Chavet, D. Gibson, J. Meyer, P. Pattanayak, A. Tomkins, and J. Zien. How to build a webfountain: An architecture for very large-scale text analytics. *IBM Systems Journal*, 43(1):64–77, 2004.

[8] H. Han, L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In *Proceedings of the 2004 joint ACM/IEEE conference on Digital libraries*, pages 296–305, 2004.

[9] R. Jones, A. McCallum, K. Nigam, and E. Riloff. Bootstrapping for text learning tasks. In *IJCAI-99 Workshop on Text Mining: Foundations, Techniques, and Applications*, 1999.

[10] G. S. Mann and D. Yarowsky. Unsupervised personal name disambiguation. In *CoNLL*, 2003.

[11] M. Meila. Comparing clusterings. Technical Report 418, UW Statistics Department, 2002.

[12] G. A. Miller. Wordnet: A lexical database. *Communications of the ACM*, 38(11):39–41, 1995.

[13] J. Novak, P. Raghavan, and A. Tomkins. Anti-aliasing on the web. In *Proceedings of the 13th international conference on World Wide Web*, pages 30–39. ACM Press, 2004.

[14] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *Advances in Neural Information Processing (NIPS)*, 2002.

[15] R.Guha and R. McCool. Tap: Towards a web of data. `http://tap.stanford.edu/`.

[16] E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1044–1049, 1996.

[17] D. A. Smith and G. Crane. Disambiguating geographic names in a historic digital library. In *Proceedings of ECDL*, pages 127–136, 2002.

[18] N. Wacholder, Y. Ravin, and M. Choi. Disambiguation of proper names in text. In *Proceedings of Fifth Conference on Applied Natural Language Processing*, pages 202–208, 1997.