Who's Who? Identifying Concepts and Entities across Multiple Documents

Zunaid Kazi and Yael Ravin
T. J. Watson Research Center, IBM, POB 704, Yorktown Heights, NY 10598
zhkazi, ravin @us.ibm.com

Abstract

A number of research and software development groups have developed technology for identifying terms and names in documents and associating them with concepts and named entities, but few have addressed coreference of concepts and entities across multiple documents in a collection. Cross-document coreference is challenging, since a collection of documents consists of multiple discourse contexts, with a many-to-many correspondence between terms and names on one hand and the concepts and entities they refer to on the other. In this paper we describe extensions to our intra-document term and name identification for coreferencing concepts and entities across documents.

1. Introduction

The need to identify and extract important terms and names in online text documents is now commonly acknowledged by researchers and practitioners in the fields of information retrieval, knowledge management and digital libraries. Since terms and names express salient concepts and entities discussed in documents, identifying them is a necessary first step towards achieving a reduction in the ever-increasing volumes of online text. There are several challenging aspects to the identification of terms and names (henceforth, commonly referred to as *terms*) in text: first, identifying the text strings (words or phrases) that express terms; then, relating terms to the concepts and entities discussed in the document; and, finally, identifying the same concepts and entities across documents.

In relating terms to concepts and entities the main difficulty is the many-to-many mapping among them. This is particularly true of names and named entities. A single entity can be referred to by several name variants: *Ford Motor Company*, *Ford Motor Co.*, or simply *Ford*. A single variant often names several entities: *Ford* refers to the car company, but also to a place (Ford, Michigan) as well as to several people (President Gerald Ford, Senator Wendell Ford, and others). Context is crucial in identifying the intended mapping. A document usually defines a single context, in which it is quite unlikely to find several entities corresponding to the same variant. For example, if the document talks about the car company, it is unlikely to also discuss Gerald Ford. Thus, within documents, the problem is usually reduced to a many-to-one mapping between several variants and a single entity.

Much recent work has been devoted to the identification of terms and names and to associating them with concepts and entities within the document; both as research efforts ([1], [2] and [3]) and as commercial software packages ([4], [5] and [6]). But few have investigated their equivalence, or coreference, across documents. In a collection of documents, there are multiple contexts; variants may or may not refer to the same concept or entity; and ambiguity is a much greater problem. One recent promising approach, discussed by Bagga and Baldwin in [7], compares all the document contexts in which similar terms appear in order to determine whether the terms refer to the same concepts. Although promising, this approach can be prohibitively expensive both in storage and in the number of n-to-n comparisons over a large collection of documents.

Our approach to this problem of cross-document coreference is to minimize the need for context comparisons as much as possible. Our algorithm for coreferencing two or more terms across documents capitalizes on the careful intra-document recognition techniques we have developed. We define compatible terms and group them as good candidates for coreference. Then we compare their internal structures to determine if they indeed corefer. Context comparison is required only for the much smaller residue of compatible terms that cannot be merged by means of the algorithms we describe here.

In Section 2, we describe our intra-document name processing and in Section 3 our extension for cross-document coreferencing of names. Section 4 describes intra-document processing of terms. Only a subset of those, single-word uppercase terms, pose a challenge across documents. Their disambiguation is described in Section 5. In Section 6 we present some evaluations of our approach. Section 7 discusses issues, problems, and future work.

2. Intra-Document name identification

Our group has developed a set of tools, called Talent, to analyze and process information in text, one of which, Nominator [8], is used to discover names. Text, as illustrated by the following excerpt from [9], is analyzed into tokens - words, tags, and punctuation elements.

```
...The professional conduct of lawyers
in other jurisdictions is guided by
American Bar Association rules ... The
ABA has steadfastly reserved ... But
Robert Jordan, a partner at Steptoe &
Johnson who took the lead in ... "The
practice of law in Washington is very
different from what it is in Dubuque,"
he said. ... Mr. Jordan of Steptoe &
Johnson ...
```

Nominator forms a list of candidate names by scanning the tokenized document and collecting (mostly) capitalized sequences, such as,

```
American Bar Association
Robert Jordan
Steptoe & Johnson
ABA
Washington
Dubuque
Mr. Jordan of Steptoe & Johnson
```

Each candidate name is examined for points (such as *of* or *&)* at which it could be split into component names and then processed by a set of heuristics to determine how to split it into smaller independent names. For example, *Mr. Jordan of Steptoe & Johnson* is split into *Mr. Jordan* and *Steptoe & Johnson*. If evidence is insufficient, we err on the conservative side and do not split. Further splitting becomes possible when more evidence from the whole collection is available, as discussed in Section 3.

Nominator links all variants referring to the same entity within the document. For example *ABA* is linked to *American Bar Association* as a possible abbreviation. Each linked group is categorized by an entity type and assigned a canonical string as identifier. The result for the sample text is shown below, with canonical strings followed by their entity type and variant names.

```
American Bar Association [ORG]: ABA
Steptoe & Johnson [ORG]
Washington [PLACE]
Dubuque [PLACE]
Robert Jordan [PERS]: Mr. Jordan
```

In a typical document, a single entity may be referred to by many name variants, which differ in their degree of potential ambiguity. To disambiguate, we identify *anchors*, variant names that unambiguously refer to certain entity types, and then link more ambiguous variants to each anchor to form equivalence groups.

A few simple indicators determine the entity type of a name, such as *Mr.* for a person or *Inc*. for an organization. More commonly, several pieces of positive and negative evidence are accumulated in order to make this judgment. Some combinations may result in a high negative score -- highly confident that this cannot be a person name. For example, *Justice Department* lacks a personal title, as well as a first name, and its last name is marked as an organization word. *Justice Johnson*, whose last name is not an organization, receives a low positive score.

Names with low or zero scores are first tested as possible variants of names with high positive scores. However, if they are incompatible with any, they are assigned a weak entity type. Thus in the absence of any other evidence in the document, *Beverly Hills* or *Susan Hills* will be classified as PERS?.

3. Cross-Document name identification

After all the documents of the collection have been processed and all the evidence is available, more splitting of compound names can be done. The heuristics for splitting names within the document [10] fail to address the case of organization names of the form *X of Y* or *X in Y*, where *Y* is a place, such as *Fox News Channel in New York City* or *Prudential Securities in Shanghai*. The intra-document heuristic splits names if their components occur on their own within the document. But it is not appropriate for cases illustrated by these examples since the short form may be licensed in the document only because the full form exists in it. We need evidence that the short form occurs by itself in other contexts. First, we sort these names and verify that there are no ambiguities. For example, it may appear that *Union Bank of Switzerland in San Francisco* is a candidate for splitting, since *Union Bank of Switzerland* occurs as a canonical name, but the existence of *Union Bank of Switzerland in New York* signals an ambiguity -- there are several distinct entities whose name starts with *Union Bank of Switzerland* and so no splitting applies. Similar ambiguity is found with *Federal District Court in New York*, *Federal District Court in Philadelphia*, etc. If the name is split, we repair the cross document statistics by folding the occurrence statistics of the combined form with those of each of the parts. Note that this definition of ambiguity is dependent on the particular names found in the

collection. For example, in the [11] collection, the only *Prudential Securities of/in …* found was *Prudential Securities in Shanghai*. Thus we take the collection to be representative of the names and concepts in the domain.

We now turn to merging canonical names from different documents that refer to the same real entity. The algorithm sorts names with common substrings from least to most ambiguous. For example, PERS names are sorted by identical last names. The least ambiguous ones also contain a first name and middle name, followed by ones containing a first name and middle initial, followed by ones containing only a first name, a first initial and finally the ones with just a last name. PERS names may also carry gender information, determined either on the basis of the first name (e.g. *Bill* but not *Jamie*) or a gender prefix (e.g. *Mr.*, but not *President*) of the canonical form or one of its variants. PLACE names are sorted by common initial strings. The least ambiguous have the pattern of <small place, big place>. By comparing the internal structure of these sorted groups, we are able to divide them into mutually exclusive sets and a residue of mergeable names. *Exclusives* have incompatible features that prevent any further merging among them. *Mergeables* are compatible with some or all of the Exclusives. Further comparisons of the internal structure indicate that some Mergeables can be aggregated with the appropriate Exclusives without any further tests. Context comparisons are needed for the much smaller residue that remains.

To illustrate with an example, here are the results of intra-document analysis for *Bush*:

```
17 docs: Bush, unspecified category or
gender.
1 doc: Christopher Bush [PERS?], male.
1 doc: Douglas Bush [PERS?], male.
26 docs: George Bush [PERS?], male
2 docs: George Bush [PERS]: President
Bush; male
1 doc: George W. Bush [PERS|: Gov.
George W. Bush, President George Bush;
male
1 doc: Mr. Bush [PERS], male
2 docs: President Bush [PERS]
7 docs: Vannevar Bush, unspecified
category or gender.
```

If we stopped here, we would have to carry out 58x58 comparisons.

In our cross-document processing we create four Exclusives:

```
E1 – Christopher Bush
E2 – Douglas Bush
E3 – George W. Bush
E4 – Vannevar Bush
```

And the following Mergeables, compatible with Exclusives as indicated:

```
M1 – George Bush, mergeable with E3
because of first name and gender
M2 - Mr. Bush, mergeable with E1-E4
M3 – President Bush, mergeable with E1-
E4
M4-9 – Bush, mergeable with E1-E4
```

The first merging we perform is between identical canonical strings that are two words or longer. Identical *single-word* canonical strings are too ambiguous for this kind of blind merging; hence we leave M4-9 separate. This merging puts 28 *George Bush*, 2 *President Bush*, and 7 *Vannevar Bush* into 3 equivalence classes. Most merges of this kind involve names of the same category (e.g., PERS) but others involve merging a weak category with a different strong one, as in *Digital City*, which is a PLACE? in one document and an ORG in another; or *Carla Hills, U.S,* PLACE? and Mrs. *Carla Hills,* PERSON.

The next merge we perform is between Mergeables and their compatible Exclusives if any of their variants share a common prefix. In this case, *President* puts E3, M1 and M3 into one equivalence class.

By grouping intra-document equivalence classes into larger equivalence classes we have reduced the number of context comparisons to 7x4, that we need to compare M2 and M4-M9 with each of E1 to E4.

The distribution of Exclusives and Mergeables varies significantly from one sorted group to another. On one hand, there are “famous” entities, such as President Bush. These tend to have at least one Exclusive with a high number of occurrences and quite a few Mergeables, as a famous entity is assumed to be part of the reader’s general knowledge and is therefore not always fully and formally introduced. A careful context comparison will be beneficial in this case. On the other end of the scale, there are non-famous entities. There may be a great number of Exclusives, especially for common last names but the frequency of occurrences is relatively low. Expensive processing may not be justified for low-frequency Exclusives. It seems that we can establish a tradeoff between processing cost versus overall accuracy gain and decide ahead of time how much disambiguation processing is required for a given application.

4. Intra-Document term identification

Semantically, names refer to entities such as individual people or specific places. Single- or multi-word terms, by contrast, refer to concepts defined or discussed in the domain covered by a collection of documents. We return to single-word terms below. Multi-word terms form a subset of the noun phrases occurring in the document. They are lexical; that is, their meaning is not usually derivable from the meanings of their component words (e.g., *central processing unit*) and since they are central to the domain, they are often repeated in the text.

The module that identifies multi-word terms, called Terminator, scans the document tokens for all sequences of words that match grammatical structures which can be captured as given in (1) below:

((A|N)+|((A|N)*(NP)?)(A|N)*)N (1)

A is an adjective; N is a noun; and P is a preposition. In words, a candidate term is a multi-word noun phrase; and it either is a string of nouns and/or adjectives, ending in a noun *(cumulative distribution function)*, or it consists of two such strings, separated by a single preposition (*degrees of freedom)* [12]. The part of speech of each word is determined by lookup in an online dictionary of English. Since many words in English have more than one part of speech, the procedure may extract sequences that are not really noun phrases, such as *price rose*. However, discarding multi-word terms that occur only once in a document removes many of these false phrases.

Terms that are found only as proper substrings of other terms are discarded. For example, *dimension space* is an invalid term, as it never occurs outside of a larger term, such as *lower-dimension space* or *multi-dimension space*.

Terminator groups together intra-document morphological variants (*linear functions* and *linear function*) and case variants. Another tool, Abbreviator, associates multi-word terms with acronyms or abbreviations that occur within the same document. As multi-word terms are seldom ambiguous, they can be safely merged across documents and we do not discuss them further here. The residue, single words (mostly nouns, and some verbs and adjectives) that are neither part of names nor of multi-word terms, are identified as single-word terms and also grouped with morphological and case variants. Unlike multi-word terms, single words are highly ambiguous and require careful examination before they can be merged with other terms across documents.

The most common source of ambiguity is uppercase single words. Consider words like *Wired* -- a name of a magazine, but also an adjective in sentence-initial position -- as well as *Bush*, *Word*, *Enliven*, and many others. In the absence of a dictionary of names -- and Talent does not assume one -- every capitalized word potentially poses this problem. Many cases of this kind of ambiguity are handled within the document. For example, if the capitalized word also occurs in lowercase in the document, we are quite confident that it is a regular word, and the uppercase form is considered a variant of the lowercase one. If the word occurs in uppercase in the middle of a sentence, we are quite sure it is a name. But if there are no lowercase occurrences and the uppercase word occurs in either in sentence-initial position(s) or in headers and titles, we cannot be sure. In these cases, the word is typed as a term, not a name, and its initial capitalization is preserved. As a result, on the collection level, we have many capitalized strings in multiple categories: single-word terms, names and name variants. In the next section we discuss how to disambiguate these.

5. Merging single words across documents

First, there is the unambiguous case, illustrated in Table 1. (Columns indicate the categories assigned to the string during intra-document analysis.) In some document(s), the string exists as a lowercase term (noun, verb or adjective), which may have uppercase variants, and in others, as an uppercase name, not further categorized as a person, place or organization. It is clear that the two forms are quite distinct and should not be merged.

Table 1 : Unambiguous Cases

Lower-case Term	**Uncategorized Name**
enliven	Enliven
word	Word
wired	Wired

Table 2 : Ambiguous Cases

Upper-case Term	**Lower-case Term**	**Uncat. Name**	**Name is Variant of**
Finds	finds	----	----
Loss	loss	----	----
Allied	----	Allied	----
Microsoft	----	Microsoft	Microsoft Corp.
N.Y.	----	----	New York

As mentioned above, ambiguity is introduced by uppercase single-word terms created during intra-document analysis when neither a lowercase form nor other uppercase occurrences exist in the document. This ambiguity is resolved on the collection level if the collection contains only the name or only the lowercase term, but not both. Consider *Finds* and *Loss*, as shown in Table 2.

Since the uppercase string does not occur as a name in the collection, we can safely assume that its capitalized occurrences are variants of the lowercase noun or verb. *Allied*, *Microsoft* and *N.Y.* are also instances of an easy (but rarer) case of merging – since there are no lowercase occurrences in the collection, it is safe to assume that the capitalized strings are in fact name occurrences which failed to be recognized as such during intra-document analysis, usually because they occur in a header or title, among other words that are all capitalized.

The ambiguity is more complex when the two types exist in the collection. We have found that single occurrences of single capitalized terms can be merged as occurrences of the corresponding names, provided that the names occur more than once in some other document in the collection. Names that occur only once per document, across all documents, are usually misanalyses. The following table makes this observation clearer.

Table 3

Upper-case Term	**Uncat. Name**	**No. of Docs**	**Range of Intra-document occs.**
Find	Find	2	1
Please	Please	8	1
Met	Met	5	2-3
Sun	Sun	12	1-3
Apple	Apple	203	1-46

The single terms *Find* and *Please* should be merged with the corresponding lowercase terms found in other documents. The « names » *Find* and *Please* should also be merged with the lowercase forms. This is because they occur only once per document (in two and eight documents respectively). By contrast, we verified that all the single occurrences of *Met*, *Sun*, and *Apple* should be merged with the corresponding names and not with the lowercase terms[1]. These examples indicate that it is not the total occurrences of a name that determines whether it is valid but rather its occurrence distribution within documents. There are more total occurrences of the « name » *Please* than there are of the name *Met*, yet *Please* is not a valid name, whereas *Met* is valid, as it occurs twice in one document and three times in the other.

This disambiguation technique is another instance where cross-document evidence can correct intra-document analysis. Although our findings are still tentative, as we are trying to verify our results over larger samples of data, they conform well to recent research on the *burstiness* of salient concepts in text; that is, the observation that important concepts are likely to be mentioned more than once in a given context. [13].

6. Evaluation

Evaluating the accuracy of the results of our work is extremely difficult. The most obvious difficulty is the need to create truth data, a manual and labor-intensive job. Errors of omission, in particular, require an effort. In addition, it is difficult to compare results across document collections, as ambiguity is likely to vary for different collection sizes, date ranges and writing styles. For example, [7] found 197 articles mentioning various John Smiths in 2-years worth (1996-97) of New York Times articles. We found only 3 articles mentioning John Smith in our collection of New York Times articles ranging over 1998. Unfortunately, due to copyright issues, we cannot exchange collections and compare results.

But even if we could, our results would vary because our initial set of names and entities to compare is not the same. [7] collect all occurrences of "John.* Smith" as possible merging candidates. We also add *Jack Smith*, as *Jack* is a known nickname for *John*, *Mr. Smith* and plain *Smith*.

Nevertheless, we have used the evaluation measures developed by [7] for our cross-document PERS coreference. Briefly, the evaluation consists of calculating recall and precision for each name occurrence in each equivalence class for a family of names, such as *Bush*, and then averaging across all such occurrences. Here are the equivalence classes for the *Bush* example:

```
E1- Christopher Bush, 1 occ
E2- Douglas Bush, 2 occs
E3- George W. Bush, 41 occs
E4- Vannevar Bush, 7 occs
```

[1] One single occurrence was found in the following headline : « Mac Clones : Cannibals or Apple Seeds ?» Note that disambiguation techniques that rely on the presence of diambiguating words in the context may be mislead here to assign the fruit reading to *Apple*, because of the presence of the word *seeds*.

```
M2- Mr. Bush, 1 occ
M4- Bush, 1 occ
M5- Bush, 1 occ
M6- Bush, 4 occs
M7- Bush, 9 occs
M8- Bush, 2 occs
M9- Bush, 11 occs
```

Truth:

```
E1- Christopher Bush, 1 occ
E2- Douglas Bush, 2 occs
E3+M2+M9- George Bush, 53 occs
E4+M4…M7- Vannevar Bush, 22 occs
M8- the rock band Bush
```

Precision for $member_i$ is calculated as: number of correct members in class containing i divided by *total number* of members in class containing i.

Recall for $member_i$ is: number of correct members in class containing i divided by number of members in *truth class* containing i.

Precision for a member of M9, for example, will be 11/11 while recall will be 11/53. Average precision and recall for *Bush* will be the sum of all the member precision and recall results divided by the total number of members in E1 through M9.

Here are our average precision and recall percentages for 15 name families:

Albright:	P = 100.00;	R = 79.91
Anderson:	P = 100.00;	R = 83.33
Berger:	P = 100.00;	R = 55.86
Black:	P = 100.00;	R = 61.11
Brown:	P = 98.50;	R = 74.98
Bush:	P = 100.00;	R = 41.81
Carter:	P = 100.00;	R = 82.03
Clinton:	P = 90.71;	R = 64.41
Gore:	P = 98.79;	R = 93.59
Hayes:	P = 100.00;	R = 96.77
Herman:	P = 100.00;	R = 51.85
Jackson:	P = 100.00;	R = 96.00
Lewis:	P = 100.00;	R = 93.55
Miller:	P = 89.40;	R = 68.04
Reagan:	P = 100.00;	R = 49.63

Average Precision: 1092.87/15 = 72.85%
Average Recall: 1477.40/15 = 98.49%

7. Discussion

We chose two kinds of name examples for the evaluation: famous entities and common last names, on the assumption that the two groups will show ambiguity problems in complementary ways. Mostly, our precision is very high. This is due to our excellent intra-document analysis and to conservative cross-document merging. One interesting exception is *Clinton*. The low precision is due mainly to intra-document ambiguity in documents that mention *President Clinton* and *Mrs. Clinton*. In many of these documents, the two merge to create a canonical female *President Clinton*. We are still unsure about how to prevent this merging without appealing to external world knowledge.

Another interesting exception to high precision is the *Miller* example. We claimed in Section 3 that it is safe to merge identical canonical strings of two or more words, and most of the time it is. Out of about 60 examples of identical strings, ranging in number of occurrences from 142 to 2, only 4 were wrongly merged. Such a low error rate is acceptable, especially since it occurs for low frequency names. However, three of the faulty merges were Millers(!) : Marvin Miller, Richard W. Miller and Robert Miller. The *Miller* family stands out since it has the largest number of name entities – 41 different individuals, while the average number of entities for our 15 examples is 11.46. Perhaps we should be more cautious in merging identical strings when the number of entities is so large.

The precision/recall tradeoff is a well-known phenomenon. In the applications we are concerned with, high precision is preferred to high recall. Consider a user interested in documents referring to President Bush. We can propose at the top of the list documents associated with *George Bush*. Our high precision will ensure that they are relevant. Further down the list we can include documents associated with the residual Mergeables that are compatible with *George Bush*, if full recall is required. Or consider another application, that of populating a database with documents indexed by their named entities. High precision means that a human reviewer should only look at the residue of Mergeables.

To increase our recall, we could invest in context comparisons. We are evaluating different schemes for implementing context comparisons within our system. (See [14] for using a *Context Thesaurus* for this purpose.)

Finally, a challenge we have to address. Collections of documents change over time as new documents are added and old ones are removed. These changes may affect the degree of ambiguity of the names and terms previously identified in the collection. We need to investigate how to turn cross-document coreference into an incremental

process. In particular, we have to figure out how to recover when a previously unambiguous name or term becomes ambiguous as the collection grows as well as update the occurrence statistics for processes such as information retrieval that depend on the results of cross-document name and concept identification.

8. References

[1]. Tipster Text Program. *Sixth Message Understanding Conference (MUC-6).*

[2]. Tipster Text Program. *Seventh Message Understanding Conference (MUC-7).*

[3]. T. Stzalkowski and J. Wang. A Self-Learning Universal Concept Spotter. In *Proceedings of COLING-96*, pages 931-936.

[4]. NetOwl Extractor Technical Overview (White Paper). *http://www.isoquest.com/*, March 1997.

[5]. http://www.inxight.com/Inxight/Home.html

[6]. http://www.software.ibm.com/data/iminer/.

[7]. A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of COLING-ACL 1998*, pages 79-85.

[8]. Y. Ravin and N. Wacholder. Extracting Names from Natural-Language Text. *IBM Research Report 20338*. 1996.

[9]. Tipster Information-Retrieval Text Research Collection', CD-ROM, NIST, Gaithersburg, Maryland.

[10]. N. Wacholder, Y. Ravin and M. Choi. Disambiguation of proper names in text. In *5th Conference on Applied Natural Language Processing*, pages 202-208, 1997.

[11]. Articles from http://www.nytimes.com/.

[12]. Justeson, J. S. and S. Katz. Technical Terminology: Some Linguistic Properties and an Algorithm for Identification in Text. In *Natural Language Engineering*, Volume 1, pages 9-27, 1995.

[13]. S. Katz. Distribution of content words and phrases in text and language modelling. In *Natural Language Engineering*, Vol. 2 No. 1, pages 15-59, Cambridge University Press.

[14]. Y. Ravin and Z. Kazi. Is Hillary Rodham Clinton the President? Disambiguating Names across Documents. In *ACL '99 Workshop on Coreference and its Applications*. 1999.