Multi-Document Person Name Resolution

Michael Ben Fleischman
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139
mbf@mit.edu

Eduard Hovy
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
hovy@isi.edu

Abstract

Multi-document person name resolution focuses on the problem of determining if two instances with the same name and from different documents refer to the same individual. We present a two-step approach in which a Maximum Entropy model is trained to give the probability that two names refer to the same individual. We then apply a modified agglomerative clustering technique to partition the instances according to their referents.

1 Intro

Artists and philosophers have long noted that multiple distinct entities are often referred to by one and the same name (Cohen and Cohen, 1998; Martinich, 2000). Recently, this referential ambiguity of names has become of increasing concern to computational linguists, as well. As the Internet increases in size and coverage, it becomes less and less likely that a single name will refer to the same individual on two different web sites. This poses a great challenge to information retrieval (IR) and question-answering (QA) applications, which often rely on little data when responding to user queries.

Another area in which referential ambiguity is problematic involves the automatic population of ontologies with instances. For such tasks, concept-instance pairs (such as *Paul Simon/pop star*) are extracted from the web, cleaned of noise, and then inserted into an already existing ontology. The process of insertion requires that concept-instance pairs that have the same referent be merged together (e.g. *Paul Simon*/*pop star* and *Paul Simon* /*singer*). Further, instances of the same name, but with different referents, must be distinguished (e.g. *Paul Simon*/*pop star* and *Paul Simon* /*politician*).

We propose a two-step approach: first, we train a maximum entropy model to generate the probability that any two concept-instance pairs refer to one and the same referent. Then, a modified agglomerative clustering technique is used to merge the most likely instances together, forming clusters that correspond to individual referents.

2 Related Work

While there has been a great deal of work on coreference resolution within a single document, little work has focused on the challenges associated with resolving the reference of identical person names across multiple documents.

Mann and Yarowsky (2003) are amongst the few who have examined this problem. They treat it as a clustering task, in which, a combination of features (such as, a weighted bag of words and biographic information extracted from the text) are given to an agglomerative clustering algorithm, which outputs two clusters representing the two referents of the query name.

Mann and Yarowsky (2003) report results on two types of evaluations: using hand-annotated webpages returned from truly ambiguous searches, they report precision/recall scores of 0.88/0.73; using “psuedonames”[1] they report an accuracy of 86.4%.

[1] Borrowing from techniques in word sense disambiguation, they create a test set of 28 “pseudonames” by ran-

While Mann and Yarowsky (2003) describe a number of useful features for multi-document person name resolution, their technique is limited by only allowing a set number of referent clusters. Further, as discussed below, their use of artificial test data makes it difficult to determine how well it generalize to real world problems.

Bagga and Baldwin (1998) also present an examination of multi-document person name resolution. They first perform within-document coreference resolution to form coreference chains for each entity in each document. They then use the text surrounding each reference chain to create summaries about each entity in each document. These summaries are then converted to a bag of words feature vector and are clustered using the standard vector space model often employed in IR.

They evaluated their system on 11 entities named John Smith taken from a set of 173 New York Times articles. Using an evaluation metric similar to a weighted sum of precision and recall they get an F-measure of 0.846.

Although their technique allows for the discovery of a variable number of referents, its use of simplistic bag of words clustering is an inherently limiting aspect of their methodology. Further, that they only evaluate their system, on a single person name begs the question of how well such a technique would fair on a more real-world challenge.

3 Maximum Entropy Model

3.1 Data

Fleischman et al. (2003) describe a dataset of concept-instance pairs extracted automatically from a very large corpus of newspaper articles. The dataset (referred to here as the ACL dataset) contains approximately 2 million pairs (of which 93% are legitimate) in which the concept is represented by a complex noun phrase (e.g. *president of the United States*) and the instance by a name (e.g. *William Jefferson Clinton*).[2]

A set of 2675 legitimate concept-instance pairs was randomly selected from the ACL dataset described above; each of these was then matched with another concept-instance pair that had an *identical* instance name, but a *different* concept name. This set of matched pairs was hand tagged by a human annotator to reflect whether or not the identical instance names actually referred to the same individual. The set was then randomly split into a training set of 1875 matched pairs (84% referring to the same individual), a development set of 400 matched pairs (85.5% referring to the same individual), and a test set of 400 matched pairs (83.5% referring to the same individual).

3.2 Features

In designing a binary classifier to determine whether two concept-instance pairs refer to the same individual, we formulate a number of different features used to describe each matched pair. These features are summarized in Table 1, and described in more detail below.

Name Features

We use a number of methods meant to express information available from the orthography of the instance name itself. The first of these features (Name-Common) seeks to estimate the commonality of the instance name. With this features we hope to capture the intuition that more common names (such as *John Smith*) will be more likely to refer to different individuals than more uncommon names (such as *Yasir Arafat*). We calculate this feature by splitting the instance name into first, middle (if necessary) and last sub-names. We then use a table of name frequencies downloaded from the US census website to give each sub-name a score; these scores are then multiplied together for a final value.

The second name statistic feature estimates how famous the instance name is. With this features we

domly selecting two names from a hand crafted list of 8 individuals (e.g., Haifa Al-Faisal and Tom Cruise) and treat the pair as one name with two referents.

[2] Although the dataset includes multiple types of named entities, we focus here only on person names.

hope to capture the intuition that names of very famous people (such as Michael Jackson) are less likely to refer to different individuals than less famous, yet equally common, names (such as John Smith). We calculate this feature in two ways: first, we use the frequency of the instance name as it appears in the ACL dataset to give a representation of how often the name appears in newspaper text (Name-Fame); second, we use the number of hits reported on google.com for a query consisting of the quoted instance name itself (Web-Fame). These fame features are used both as is and scaled by the commonality feature described above.

Web Features

Aside from the fame features described above, we use a number of other features derived from web search results. The first of which, called WebIntersection, is simply the number of hits returned for a query using the instance name and the heads of each concept noun phrase in the match pair; i.e., (name + head1 +head2).

The second, called WebDifference, is the absolute value of the difference between the hits returned from a query on the instance name and just the head of concept 1 vs. the instance name and just the head of concept 2; i.e., abs ((name + head1) -(name +head2)).

The third, called WebRatio, is the ratio between the WebIntersection score and the sum of the hits returned when querying the instance name and just the head of concept 1 and the instance name and just the head of concept 2; i.e., (name + head1 +head2) / ((name + head1) +(name +head2)).

Overlap Features

In order to capture some aspects of the contextual cues to referent disambiguation, we include features representing the similarity between the sentential contexts from which each concept-instance pair was extracted. The similarity metric that we use is a simple word overlap score based on the number of words that are shared amongst both sentences. We include scores in which each non-stop-word is treated equally (Sentence-Count), as well as, in which each non-stop-word is weighted according to its term frequency in a large corpus (Sentence-TF). We further include two similar features that only examine the overlap in the concepts (Concept-Count and Concept-TF).

Name Features	
Feature Name	*Description*
Name-Common	frequency of name in census data
Name-Fame	frequency of name in ACL dataset
Web-Fame	# of hits from name query
Web Features	
Web Intersection	query(name + head1 +head2)

Web Difference	abs(query(name + head1)
	+ query(name +head2))

Web Ratio	query(name + head1 +head2) / (qry(name + head1) +qry(name +head2))
Overlap Features	
Sentence-Count	# of words common to context of both instances
Sentence-TF	as above but weighted by term frequency
Concept-Count	# of words common to concept of both instances
Concept-TF	as above but weighted by term frequency
Semantic Features	
JCN	sem. dist. of Jiang and Conrath
HSO	sem. dist. of Hirst and St. Onge
LCH	sem. dist. of Leacock and Chodrow
Lin	sem. dist. of Lin
Res	sem. dist. of Resnik
Estimated Statistics	
F1	p(i1=i2 | i1→A, i2→B)
F2	p(i1→A, i2→B | i1=i2)
F3	p(i1→A | i2→B) + p(i2→B | i1→A)
F4	p(i1→A, i2→B) / (p(i1→A) + p(I2→B))

Table 1. Features used in Max. Ent. model split according to feature type.

Semantic Features

Another important clue in determining the coreference of instances is the semantic relatedness of the concepts with which they are associated. In order to capture this, we employ five metrics described in the literature that use the WordNet ontology to determine a semantic distance between two lexical items (Budanitsky and Hirst. 2001). We use the implementation described in Pedersen (2004) to create features corresponding to the scores on the following metrics shown in Table 1. Due to problems associated with word sense ambiguity, we take the maximum score amongst all possible combinations of senses for the heads of the concepts in the matched pair. The final output to the model is a single similarity measure for each of the eight metrics described in Pedersen (2004).

Estimated Statistics Features

In developing features useful for referent disambiguation, it is clear that the concept information to which we have access is very useful. For example, given that we see *John Edwards /politician* and *John Edwards /lawyer*, our knowledge that politicians are often lawyers is very useful in judging referential identity.[3] In order to exploit this information, we leverage the strong correlation between orthographic identity of instance names and their referential identity.

As described above, approximately 84% of those matched pairs that had identical instance names referred to the same referent. In a separate examination, we found, not surprisingly, that nearly 100% of pairs that were matched to instances with *different* names (such as *Bill Clinton* vs. *George Clinton*) referred to different referents.

We take advantage of this strong correlation in developing features by first making the (admittedly wrong) assumption that orthographic identity is equivalent to referential identity, and then using that assumption to calculate a number of statistics over the large ACL dataset. We postulate that the noise introduced by our assumption will be offset by the large size of the dataset, yielding a number of highly informative features.

The statistics we calculate are as follows:

P1: The probability that instance 1 and instance 2 have the same referent given that instance 1 is paired with concept A and instance 2 with concept B; i.e., p(i1=i2 | i1→A, i2→B)

P2: The probability that instance 1 is paired with concept A and instance 2 with concept B given that instance 1 and instance 2 have the same referent; i.e., p(i1→A, i2→B | i1=i2)

P3: The probability that instance 1 is paired with concept A given that instance 2 is paired with concept B *plus* the probability that instance 2 is paired with concept B given that instance 1 is paired with concept A; i.e., p(i1→A | i2→B) + p(i2→B | i1→A)

P4: The probability that instance 1 is paired with concept A and instance 2 is paired with concept B divided by the probability that instance 1 is paired with concept A plus the probability that instance 2 is paired with concept B; i.e., p(i1→A, i2→B) / (p(i1→A) + p(i2→B))

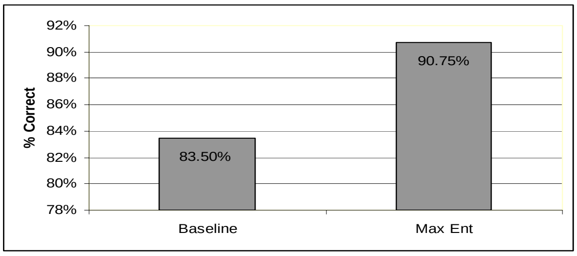

Figure 1. Results of Max. Ent. classifier on held out test data compared to baseline (i.e., always same referent).

Aside from the noise introduced by the assumption described above, another problem with these features arises when the derived probabilities are based on very low frequency counts. Thus, when adding these features to the model, we bin each feature according to the number of counts that the score was based on.

3.3 Model

Maximum Entropy (Max. Ent.) models implement the intuition that the best model will be the one that is consistent with the set of constrains imposed by the evidence, but otherwise is as uniform as possible (Berger et al., 1996). We model the probability of two instances having the same referent (*r*=[1,0]) given a vector of features *x* according to the Max. Ent. formulation below:

$$p(r \mid x) = {}^{1}\!/_{Z_x} \exp[\sum_{i=0}^{n} \lambda_i f_i(r,x)]$$

Here Z_x is a normalization constant, $f_i(r,x)$ is a feature function over values of *r* and vector elements, *n* is the total number of feature functions, and λ_i is the weight for a given feature function. The final output of the model is the probability

[3] It should be noted that this feature is attempting to encode knowledge about what concepts occur together in the real world, which is different than, what is being encoded in the semantic features, described above.

given a feature vector that *r*=1; i.e., the probability that the referents are the same.

We train the Max. Ent. model using the YASMET Max. Ent. package (Och, 2002). Feature weights are smoothed using Gaussian priors with mean 0. The standard deviation of this distribution is optimized on the development set, as is the number of training iterations and the probability threshold used to make the hard classifications reported in the following experiment.

3.4 Experimental Results

Results for the classifier on the held out test set are reported in Figure 1. Baseline here represents always choosing the most common classification (i.e., instance referents are the same). Figure 2 represents the learning curve associated with this task. Figure 3 shows the effect on performance of incrementally adding the best feature set (as determined by greedily trying each one) to the model.

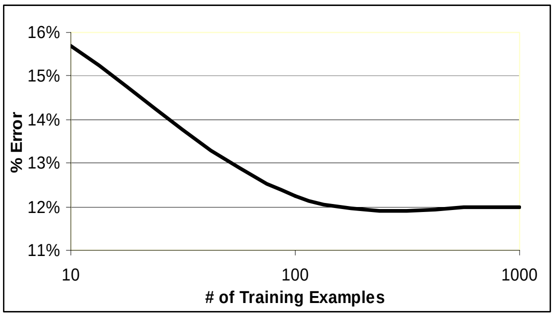

Figure 2. Learning curve of Max. Ent. model.

3.5 Discussion

It is clear from the results that this model outperforms the baseline for this task (p>0.01) (p<0.01) (Mitchell, 1997). Interestingly, although the number of labeled examples that were used to train the system was by no means extravagant, it appears from the learning curve that increasing the size of the training set will not have a large effect on classifier performance. Also of interest, Figure 3 shows that the greedy feature selection technique found that the most powerful features for this task are the estimated statistic features and the web features. While the benefit of such large corpora features is not surprising, the relative lack of power from the semantic and overlap features (which exploit ontolological and contextual information) was surprising. [4] In future work, we will examine how more sophisticated similarity metrics and larger windows of context (e.g., the whole document) might improve performance.

4 Clustering

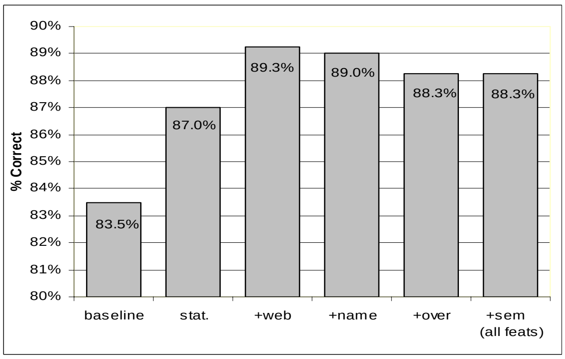

Figure 3. Results of Max. Ent. classifier on held out data using different subsets of feature types. Feature types are greedily added one at a time, starting with Estimated Statistics and ending with Semantic Features.

Having generated a model to predict the probability that two concept-instance pairs with the same name refer to the same individual, we are faced with the problem of using such a model to partition all of our concept-instance pairs according to the individuals to which they actually refer. Although, ideally, we should be able to simply apply the model to all possible pairs, in reality, such a methodology may lead to a contradiction.

For example, given that the model predicts instance A is identical to instance B, and in addition, that instance B is identical to C, because of the transitivity of the identity relation, we must assume that A is identical to C. However, if the model predicts that A is *not* identical to C, (which can and does occur) we must assume the model is wrong in at least one of its three predictions.

[4] Note that for these tests, the model parameters are not optimized for each run; thus, the performance is slightly worse than in Figure 1.

Following Ng and Cardie (2002), we address this problem by clustering each set of concept-instance pairs with identical names, using a form of group-average agglomerative clustering, in which the similarity score between instances is just the probability output by the model. Because standard agglomerative clustering algorithms are $O(n^3)$ if cosign similarity metrics are not used (Manning and Schutze, 2001), we adapt the method to our framework. Our algorithm operates as follows[5]:

```
On input D={concept-instance pairs of same name}, build a fully connected graph G with vertex set D:
1) Label each edge (d,d') in G with a score corresponding to the probability of identity predicted by the Max. Ent. model
2) While the edge with max score in G > threshold:
   a. Merge the two nodes connected by the edge with the max score.
   b. For each node in the graph
      a. Merge the two edges connecting it to the newly merged node
      b. Assign the new edge a score equal to the avg. of the two old edge scores.
```

The final output of this algorithm is a new graph in which each node represents a single referent associated with a set of concept-instance pairs. This algorithm provides an efficient way, $O(n^2)$, to compose the pair-wise information given by the model. Further, because the only free parameter is a merging threshold (which can be determined through cross-validation) the algorithm is free to choose a different number of referents for each instance name it is tested on. This is critical for the task because each instance name can have any number of referents associated with it.

4.1 Test Data

In order to test clustering, we randomly selected a set of 31 instance names from the ACL dataset, 11 of which referred to multiple individuals and 20 of which had only a single referent[6]. Each concept-instance pair with that instance name was then extracted and hand annotated such that each individual referent was given a unique identifying code.

We chose not to test on artificially generated test examples (such as the pseudo-names described in Mann and Yarowsky, 2003) because of our reliance on name orthography in feature generation (see section 3.2). Further, such pseudo-names ignore the fact that names often correlate with other features (such as occupation or birthplace), and that they do not guarantee clean test data (i.e., the two names chosen for artificial identity may themselves each refer to multiple individuals).

4.2 Experimental Design

In examining the results of the clustering, we chose to use a simple clustering accuracy as our performance metric. According to this technique, we match the output of our system to a gold standard clustering (defined by the hand annotations described above).[7]

We compare our algorithm on the 31 sets of concept-instance pairs described above against two baseline systems. The first (baseline1) is simply a single clustering of all pairs into one cluster; i.e., all instances have the same referent. The second (baseline2) is a simple greedy clustering algorithm that sequentially adds elements to the previous cluster whose last-added element is most similar (and exceeds some threshold set by cross validation).

4.3 Results

In examining performance, we present a weighted average over these 31 instance sets, based on the number of nodes (i.e., concept-instance pairs) in each set of instances (total nodes = 1256). Cross-validation is used to set the threshold for both the baseline2 and modified agglomerative algorithm.

[5] This algorithm was developed with Hal Daume (technical report, in prep.).

[6] In an examination of 113 different randomly selected instance names from the ACL dataset we found that 32 appeared only once in the dataset, 53 appeared more than once but always referred to the same referent, and 28 had multiple referents.

[7] While this is a relatively simple measure, we believe that, if anything, it is overly conservative, and thus, valid for the comparisons that we are making.

These results are presented in Table 2. Figure 4 examines performance as a function of the number of referents within each of the 31 instance sets.

4.4 Discussion

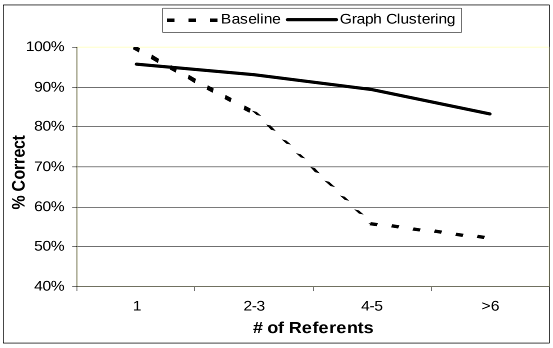

Figure 4. Plot of performance for modified agglomerative clustering and Baseline system as a function of the number of referents in the test set.

While the algorithm we present clearly outperforms the baseline2 method over all 31 instance sets (p<0.01), we can see that it only marginally outperforms our most simple baseline1 method (p<0.10) (Mitchell, 1997). This is due to the fact that for each of the 20 instance sets that only have a single referent, the baseline achieves a perfect score, while the modified agglomerative method only achieves a score of 96.4%. Given this aspect of the baseline, and the distribution of the data, the fact that our algorithm outperforms the baseline at all speaks to its usefulness for this task.

A better sense of the usefulness of this algorithm, however, can be seen by looking at its performance only on instance sets with multiple referents. As seen in Table 3, on multiple referent instance sets, modified agglomerative clustering outperforms both the baseline1 and baseline2 methods by a statistically significant margin (p<0.01) (Mitchell, 1997).

5 Conclusion

The problem of cross-document person name disambiguation is of growing concern in many areas of natural language processing. We have presented a two-step methodology for the disambiguation of automatically extracted concept-instance pairs. Our approach first applies a Maximum Entropy model to all concept-instance pairs that share the same instance name. The output probabilities of this model are then inputted to a modified agglomerative clustering algorithm that partitions the pairs according to the individuals to which they refer. This algorithm not only allows for a dynamically set number of referents, but also, outperforms two baseline methods.

A clear example of the success of this algorithm can be seen in the output of the system for the instance set for *Michael Jackson* (Appendix A, Table 2). Here, a name that refers to many individuals is fairly well partitioned into appropriate clusters. With the instance set for *Sonny Bono* (Appendix A, Table 1), however, we can see why this task is so challenging. Here, although, *Sonny Bono* only refers to one individual, the system finds (like many of the rest of us) that the likelihood of a singer also being a politician is so low that the name must refer to two different people. While this assumption is often true (as is the case with *Paul Simon*), we would have hoped that information from our web and fame features would have overridden the system’s bias in this circumstance.

In future work we will examine how other features may be useful in attacking such hard cases. Also, we will examine how this technique can be applied more generally to problems that exist between non-identical, but similar names (e.g. *Bill Clinton* vs. *William Jefferson Clinton*).

Acknowledgements

The authors would like to thank Regina Barzilay for her helpful comments and advice, and Hal Daume for his useful insights into and discussion of the problem. We also thank Deb Roy for his continued support

.

References

A. Bagga, and B. Baldwin. 1998. *Entity-Based Cross-Document Coreferencing Using the Vector Space Model.* . COLING-ACL'98.

A. Berger, S. Della Pietra and V. Della Pietra, 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, vol. 22, no. 1.

A. Budanitsky and G. Hirst . 2001. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In Workshop on WordNet and Other Lexical Resources, NAACL.

J. Cohen and E. Cohen. 1998. *The Big Lebowski*. Film. Columbia TriStar Pictures.

M. Fleischman, E. Hovy, and A. Echihabi. 2003. *Offline Strategies for Online Question Answering: Answering Questions Before They Are Asked.* ACL, Sapporo, Japan.

G. S. Mann, David Yarowsky, 2003 *Unsupervised Personal Name Disambiguation*, CoNLL, Edmonton, Canada.

C. Manning and H. Schutze. 2001 *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Ma.

A.P. Martinich, ed. 2000. *The Philosophy of Language*, Oxford University Press, Oxford, UK.

T. Mitchell. 1997. Machine Learning. McGraw-Hill International Editions, New York, NY. Pgs. 143-145.

V. Ng and C. Cardie Improving Machine Learning Approaches to Coreference Resolution. *ACL*, 2002.

F.J. Och. 2002. Yet another maxent toolkit: YASMET. www-i6.informatik.rwth-aachen.de/Colleagues/och/.

T. Pedersen. 2004. WordNet::Similarity v0.07 . http://www.d.umn.edu/~tpederse/similarity.html.

Appendix A. *Sample Cluster Output*.

Cluster 1
time pop star
the singer
onetime singer
former singer
pop singer
former entertainer
former pop star
former pop singer
entertainer
onetime beau
Singer

Cluster 1 (cont)
former rock star
Cluster 2
Lawmaker
crooning lawmaker
mayoral candidate
republican politician
congressman
Cluster 3
A freshman republican

Table 1. Output clusters for the name *Sonny Bono.*

Cluster 1
platinum recording artist
cbs records artist
artist
Cluster 2
singer
pop idol
day pop superstar
international pop star
starring singer
american singer
rock superstar
suing pop superstar
pop superstar
enigmatic pop superstar
featuring pop star
embattled pop star
controversial pop star
including singer
featuring singer
even singer
signing pop performer
pop singer
surrounding entertainer
joining entertainer
including entertainer
singing superstar
including superstar
american superstar
superstar
ailing superstar
reuter pop superstar
reclusive pop superstar
quiet pop superstar
embattled pop superstar
alleging pop superstar
music superstar
the us pop star
rock star
pop star
entertainer
pop recording star
newlywed pop star
fellow pop star
the singer
superstar singer
setting singer
rock singer
surrounding pop singer
suing pop singer
reuter pop singer
prague pop singer
pop singer
rock sensation
music sensation
pop sensation

Cluster 2 (cont)
rocker
american pop superstar
visiting idol
idol
pop music superstar
package entertainer
another entertainer
american pop singer
Cluster 3
local talk radio personality
kabc radio talk show host
los angeles radio personality
veteran kabc radio talk show host
ubiquitous radio commentator
radio broadcaster
broadcaster
Cluster 4
author
british beer guru
beer expert
Cluster 5
mannequin collector
Cluster 6
kfor commander
the commander of kfor
commander of kfor
british commander
Cluster 7
the nato commander of the kosovo liberation force
Cluster 8
Designer
Cluster 9
deputy secretary of transportation
deputy secretary of the department of transportation
Cluster 10
historian
Cluster 11
education department spokesman
company spokesman
dow corning spokesman
Cluster 12
judge
Cluster 13
receiver
career browns receiver
trading receiver
ravens receiver
baltimore wide receiver
agent wide receiver
wide receiver
Cluster 14
baylor offensive tackle
Cluster 15
beer writer

Table 2. Output clusters for the name *Michael Jackson*