interspeech 2008 incorporating SST 08

Unsupervised Learning of Edit Parameters for Matching Name Variants

Dan Gillick[1,2], *Dilek Hakkani-Tür*[2], Michael Levit[2]

[1]Department of Computer Science, University of California Berkeley
[2]International Computer Science Institute, Berkeley, CA
dgillick,dilek,levit@icsi.berkeley.edu

Abstract

Since named entities are often written in different ways, question answering (QA) and other language processing tasks stand to benefit from entity matching. We address the problem of finding equivalent person names in unstructured text. Our approach is a generalization of spelling correction: We compare to candidate matches by applying a set of edits to an input name. We introduce a novel unsupervised method for learning spelling edit probabilities which improves overall F-Measure on our own name-matching task by 12%. Relevance is demonstrated by application to the GALE Distillation task.
Index Terms: equivalent names, entity matching, unsupervised learning

1. Introduction

A typical question answering scenario involves a short question about a named entity and a set of documents that may contain an answer. Since proper names appearing in print vary greatly (see Table 1), robust name matching is important for finding relevant passages. We say that two names are equivalent if, lacking contextual information, a human would assume they refer to the same entity. Though there are times when context is available, allowing us to distinguish Michael Jordan the basketball player from Michael Jordan the computer scientist, we assume no such information, leaving the combination of our system with some cross-document coreference as future work.

Name	Variant	Edit Type
Abu Musab al-Zarqawi	Abu Masab al-Zarwaqi	typo
Abu Musab al-Zarqawi	al-Zarqawi	specificity
Abu Musab al-Zarqawi	Abu Musab al-Zarkawi	phonetic
Abu Musab al-Zarqawi	Adu Muhsab Zarqawwi	combination

Table 1: *Edit varieties*

While this problem is often addressed in a database setting where entities are stored in structured fields (see [1], [2]), we consider its application to unstructured text. As such, this is largely a problem of phonetic matching between strings, and our approach draws inspiration from statistical edit-based spell checking, developed in [3] and improved in [4]. We compare a given name to a set of candidates using a generalized edit distance. Then, rather than treating the edit distance itself as a score, we assign match probabilities based on learned edit likelihoods.

The novelty of our approach is twofold. First, we formulate a generalized set of structural edits for names (beyond simple character-level edits). Second, we learn probabilities for edits without labeled data by exploiting contextual similarities between structurally similar names in a large corpus.

Section 2 introduces our data, including a test set we have manually annotated. Section 3 describes our edit model and how we learn its parameters. Section 4 gives experimental results on our test set, and section 5 shows results on a practical application: the GALE information distillation task.

2. Data

For our empirical analysis, we consider two principal data sets. The first is the English newswire portion of LDC's Topic Detection and Tracking corpus TDT5, collected from 8 news sources from April to September of 2003. The second is a much smaller set of English newswire and English blog posts collected for the GALE Year-2 (Y2) evaluation (see [5]). As the Y2 corpus includes documents from a variety of sources including TDT5, the overlapping documents have been removed from our TDT5 set. See Table 2 for corpus statistics.

As our algorithms are based on named entities, we preprocess the data with a named entity recognizer. We use New York University's JET system described in [6], which performs both named entity recognition and coreference resolution. We extract only the first mention of each entity in a document as input to our system since this is usually the most formal and specific. Rarely is the first mention a pronoun, which we would hope to link to the desired name through coreference rather than name matching.

Corpus	Documents	Unique PER Names
TDT5	275,473	236,922
Y2	3,395	6,653

Table 2: *Datasets: person names (PER) are automatically extracted from news-related documents*

2.1. Human Labels for Equivalent Names

To assess precision and recall for our algorithms, we manually labeled pairs of non-identical names in the Y2 set. A pair of names was labeled *unmatched*, *matched*, or *ambiguous* (see Table 3 for examples). The judgements, particularly between *matched* and *ambiguous*, are subjective. Agreement, however, between two human labelers was high: the kappa statistic (see [7]) for inter-annotator agreement was 0.89. Mismatched labels were reviewed and finalized by the labelers. 803 name pairs are labeled as *matched*, 378 are labeled *ambiguous*, and the remainder are labeled *unmatched* by default. To make the labeling task feasible, we started with all pairs of names sharing some minimal similarity. Thus it is possible that there are some name pairs missing from the final labeled set.[1]

2.2. Experimental Setup

For the sake of evaluation, we jack-knife the Y2 corpus as follows: Take the first name as our query and find all its matches in the remainder of the corpus; then, repeat for each name. In this way, we attempt to recreate our manually labeled list. While it

[1]The names and manually labeled matches are available at http://www.icsi.berkeley.edu/ dgillick/names/index.html. While pairs of matched names can be found on the internet (e.g. Wikipedia), our test set is valuable because it allows us to measure precision and recall, and we welcome other research comparing results with ours.

10.21437/Interspeech.2008-77

may make more sense to think of cliques of equivalent names in a large, mostly disconnected forest, it is easier to evaluate pairs of names. In evaluation, we ignore ambiguous matches: they are neither penalized nor rewarded. We compute precision and recall at a variety of thresholds. F-Measure is calculated as the equally weighted harmonic mean of precision and recall:

$$FM = \frac{2 \times recall \times precision}{precision + recall} \quad (1)$$

Name 1	Name 2	Label
Sergio Viera de Mello	Annie de Mello	*unmatched*
Sergio Viera de Mello	Sergo de Mello	*matched*
Sergio Veira de Mello	de Mello	*ambiguous*

Table 3: *Labeling examples*

3. Algorithm

Though we are interested in matching names solely on the basis of their structural similarity, we show here how, in the absence of training data, we can bootstrap contextual similarity to train structural probabilities. This section describes each of the algorithmic components and how they fit together.

3.1. Structural Similarity

We preprocess all names by removing non-alphabetic characters, lowercasing, and splitting into words ("al-Qaeda" becomes ("al", "qaeda"). To compare two names, we find the minimum edit distance between the longer name, N, consisting of words $(w_1, ..., w_j)$ and the shorter name. The available edits are defined as follows (example edits appear in Table 4):

- $sub(p, q)$: Substitute character q for character p in exactly 1 word in N. Note that the character alphabet contains the null string $\emptyset$ so that $sub(p, \emptyset)$ is a character deletion and $sub(\emptyset, q)$ a character insertion. Substitution edits are only performed on words longer than some minimum length.
- $del(x, y)$: From N with x words, delete y possibly non-contiguous words.
- $abbr(x, y)$: From N with x words of some minimum length, abbreviate y of them. Abbreviation turns a word into its first letter. The minimum length restriction is to avoid abbreviation edits overlapping with letter deletions.
- $join(x, y)$: From N with x words, join y pairs of adjacent words.

Name 1	Name 2	Edit
noordin top	noorden top	sub('i','e')
noordin mohammed top	noordin top	$del(3, 1)$
noordin mohammed top	nordin m top	$abbr(3, 1)$
noor din top	noordin top	$join(3, 1)$

Table 4: *Edit Examples. Note that* sub *edits operate at the character level and the rest at the word level.*

While the standard spelling correction framework is based on proper spellings and misspellings, we have only matched pairs of names. As a result, insertions and deletions, both at the character and word level, are the same. Our model does not permit word level substitutions, however, as this would allow too much flexibility. Note that Levenshtein edit distance is often extended to include transposition edits (such as in "recieve" and "receive"), though in our experiments, adding these edits had a negligible effect.

3.2. Contextual Similarity

Taking a step back from our context-free problem for a moment, we make the observation that besides structural similarities, equivalent names tend to appear in similar contexts. To decide, given some context, whether two names are similar, we take a standard IR approach: For a name N, we construct a "document" consisting of all the sentences in the corpus that contain N. To measure the distance between two names, N_1 and N_2, we compute the semantic distance between their context documents as follows:

1. Compute the TFIDF[2] values for each stemmed word in the context document for N_1.
2. Create a vector C of the largest k values.
3. L2-normalize C to get C_1. That is, the sum of the squared TFIDF values should be 1.
4. Repeat steps 1-3 to get C_2 from N_2.
5. Take the inner product $C_1 \cdot C_2$ to get a similarity score between 0 and 1.

This method gives the distance between two vectors as the cosine of the angle between them. This particular implementation is inspired by [8], but has been simplified for our purposes[3]. A more sophisticated similarity measure based on Latent Semantic Analysis (LSA) as described in [9], which maps words to a semantic feature space, would likely improve these results.

3.3. Learning Edit Probabilities

Returning now to our question answering setup, we would like to learn the probability that two names are equivalent given the edits used to turn one into the other. If we had labeled data, that is, pairs of names differing by one edit labeled as matched or unmatched, the estimation would be straightforward.

We define $N_i \equiv N_j$ as name N_i equivalent to name N_j. Let E be any of the edits described in section 3.1. We could estimate the likelihood of an edit $P(E|N_i \equiv N_j)$, along with the prior $P(N_i \equiv N_j)$ from our training data via maximum likelihood. Note that we can separate the estimation of each edit type since we never allow two edit types to produce the same match. For example, there is no way to apply the character substitution edit to N_i to get N_j if we can get N_j from N_i via a word deletion edit. By application of Bayes Rule, we can compute the desired posteriors, $P(N_i \equiv N_j|E)$, for new potential matches as:

$$\frac{P(N_i \equiv N_j)P(E|N_i \equiv N_j)}{P(N_i \equiv N_j)P(E|N_i \equiv N_j) + P(N_i \not\equiv N_j)P(E|N_i \not\equiv N_j)} \quad (2)$$

Then, assuming independence between edits, we can compute posterior probabilities for matches that differ by multiple edits by taking the product over individual likelihoods.

This all sounds nice, but unfortunately we have no such labeled data. Instead, consider the following bootstrap procedure:

1. Search for matches in a large data set that differ by exactly one edit as in section 3.1.
2. Score each of these candidate matches using the context algorithm of section 3.2.
3. Consider any score greater than some threshold t_1 a positive example, and any score less than t_2 a negative example.
4. Estimate edit likelihoods as described above.

[2]Term Frequency Inverse Document Frequency approximates the importance of a term based on its frequency in this document, normalized by its frequency in a large and varied corpus.

[3]Following [8], we use $k = 50$.

Why might this work? The answer is twofold: First, the context scores are positively correlated with the true labels. Second, whatever errors the context algorithm makes, they are likely independent of the structural errors. This technique draws inspiration from two-stage regression as described in [10]. The correlation means we have a chance to learn something better than random guessing. This is demonstrably true. The independence means that what we learn will not be biased by the particular matches in the training set. This is an assumption which is not entirely true: two Arabic names are more likely to have matching contexts than an Arabic name and a Chinese name, suggesting that contextual similarity might be correlated with structural similarity in some cases. If the correlation is minimal however, things might work out anyway.

3.4. Training Details

As mentioned above, we train by assigning a negative label to low context scores and a positive label to large context scores. In practice, a significant proportion of matches have context score 0. These we assign to the negative class. For the positive class we tried two variants: (1) All non-zero matches are assigned to the positive class, and (2) all non-zero matches are assigned to the positive class, weighted by their context score. If the context scores themselves have some meaning beyond the zero/non-zero distinction, (2) ought to give better estimates, and it does. The maximum FM for (1) is some 3% absolute worse than (2).

A second issue is the estimation of prior probabilities. While we can estimate priors from the training corpus, these are very poor estimates for the much smaller test set. To understand why, recall that our training sample space consists of pairs of names that differ by one edit. All the probabilities we learn are in essence conditional on this fact. The prior probability, which we have referred to as $P(N_i \equiv N_j)$, is really conditional on N_i and N_j differing by one edit. In the large training set, the number of names differing by one edit as a fraction of the total number of names is significantly greater than in the smaller test set. This is because the larger set is "denser" than the smaller set, and as a result, the priors estimated from the training set likely under-estimate the true priors in the test set. Instead, we simply use uniform priors. A cheating experiment using the true test set priors yields no improvement.

3.5. The Metaphone Algorithm

Training sets for spelling correction show that some 98% of misspelled words are within two characters edits of the intended word. Names, however, are longer and more variable, so two character edits are often insufficient. Further, the matching problem is largely phonetic and English letters are only rough approximations of phonetic content.

Thus, we map each name to a phonetic space using the Metaphone Algorithm [11]: a simple rule-based method that essentially converts a string into a series of consonant sounds. "Mohammed", "Mohamed", and "Mohammad" all map to the same Metaphone string "MHMT"[4]. While there is certainly information in the vowels, they are the primary source of variability, so the distance between Metaphone mappings of names is substantially reduced. Future work involves higher quality phonetic mapping and extension to n-best mappings and lattices.

4. Experiments

Before examining results, let us clarify our experimental procedure.

1. Train structural parameters
   (a) Search for potential name matches in a large training set differing by one structural edit.
   (b) Use context algorithm to score each match.
   (c) Treating low scores as negative examples and high scores as positive examples, train structural parameters.
2. Simulate QA problems for evaluation
   (a) Extract named entities from a corpus, keeping only the first mention of each name in a document.
   (b) Clean names and split into words. Map each word to its Metaphone representation.
   (c) For each name, find all matches within k edits, then score with edit posteriors computed from learned parameters.

[4] We indicate metaphone symbols with capital letters

4.1. Structural Edits

We begin with an analysis of our edits. Table 5 shows results on our test set using single edits. Recall, which cannot be improved by learned parameters or the addition of context information, is at most 85%. What have we missed? Of the 123 missed matches, 38 differ by a del and a sub and 31 differ by two sub edits. Including these deeper edits, however, gives much poorer overall performance, though the learned parameters manage to improve FM from 0.22 to 0.49. The remaining missed matches fall mostly into two categories: word substitutions ("Alexandros Yiotopoulos" $\equiv$ "French-born Yiotopoulos") and nicknames ("Richard Gephardt" $\equiv$ "Dick Gephardt"). Simply including a list of 80 common nicknames improves recall by 5% absolute. For clarity, we do not include nicknames in this section's results, though they are included when we generate equivalent names for applications in the following section.

Edit	Untrained		Trained (Best)		FM gain
	precision	recall	precision	recall	
sub	0.51	0.46	0.74	0.42	+12%
del	0.75	0.37	0.79	0.36	+1%
$join$	0.90	0.01	0.90	0.01	+0%
$abbr$	0.90	0.01	0.90	0.01	+0%
All	0.60	0.85	0.76	0.80	+12%

Table 5: *Structural edit results on the Y2 test set. Results are shown at the maximum FM. Overall FM is 0.70 and improves to 0.78 with trained probabilities.*

4.2. Learned Edits

The learned parameters clearly improve performance, especially the sub edit parameters. What exactly have we learned? The table shows, for example, that deleting a 'B' is nearly four times more costly than deleting an 'H'. Substituting an 'M' for an 'N' is much less costly than substituting 'K' for 'N'. Such results make good sense: 'N' and 'M' are phonetically much closer than 'N' and 'K'. Similarly, more deleted words results in lower match certainty. The posterior $P(N_1 \equiv N_2 | abbr(2,2))$ is so small because two-letter abbreviations do not contain enough information: 'Nathan Young' $\not\equiv$ 'N.Y.'.

Edit	$\phi(N_i \equiv N_j)$	Edit	$\phi(N_i \equiv N_j)$
sub('B',$\emptyset$)	0.08	sub('H',$\emptyset$)	0.31
sub('N','K')	0.17	sub('N','M')	0.42
$del(4,1)$	0.78	$del(4,2)$	0.41
$join(3,1)$	0.42	$join(4,1)$	0.94
$abbr(3,1)$	0.78	$abbr(2,2)$	0.03

Table 6: *Posterior probabilities ϕ for some example edits.*

5. Application to Distillation

Matching equivalent names has many potential applications. To directly demonstrate the utility of this work, we show how we can improve the performance of an information distillation engine. The task can be summarized as follows: Retrieve sentences or phrases in a large corpus (including text and audio sources in multiple languages) relevant to a given templated query (see Table 7). Typically, audio is transcribed using automatic speech recognition and non-English documents are translated into English via machine translation. For each template, there are 15-30 problems, and the results reported here are averages over these.

The distillation system described in [12] trains a classifier to identify a sentence as relevant or irrelevant based on layers of sentence-level features. Some of the most salient features pertain to the template slots, which are often filled with named entities. As a result, robust name matching is very important: a user who fills a query slot with "Al Qaeda" is most likely interested in results pertaining to the alternate spelling "al-Qaida". Our lists of names are used to make these matches, and the resulting slot features are weighted by the scores we provide. While this paper is primarily concerned with person names, our named entity software labels organizations and locations as well. Using all of the parameters from our experiments with person names, we enumerate matches for these classes of entities as well.

We run the system with and without equivalent names for each of five templates using word-bigram and slot features. Table 7 shows the specific queries and table 8 shows our results. The most significant improvement is in template 12, which specifically contains a person name in its slot. Some of the other templates involve more complex entities like events or crimes, which may include named entities. Template 16, for example, is focused on locations, which are likely less variable than person names.

#	**Query Text**
1	List facts about event: [EVENT]
8	Describe the prosecution of [PERSON] for [CRIME]
12	Provide a biography of [PERSON]
15	Identify persons arrested from [ORGANIZATION] and give their name and role in the organization and time and location of arrest
16	Describe attacks in [LOCATION] giving location, date and number of dead and injured

Table 7: *Query templates used in evaluation. This is a subset of all the GALE Distillation templates.*

#	**Max F-Measure**		**Rel. Gain**
	Without Names	With Names	
1	0.42	0.42	+0%
8	0.46	0.50	+9%
12	0.36	0.44	+22%
15	0.43	0.44	+2%
16	0.47	0.46	-2%

Table 8: *Effect of equivalent names lists on Distillation*

6. Conclusion

We have described a system for identifying equivalent names given an input name and a list of possible matches, and specified a model for assigning probabilities to matches that we can train using unlabeled data. On our own labeled test set, we can achieve F-Measure 0.78, and can select thresholds allowing either high precision or high recall without significant loss in performance, depending on the application. We have also shown how equivalent names can improve performance in Information Distillation.

Throughout, we hinted at a few promising future directions. First, LSA can be used to improve the context scores, which ought to improve the estimated likelihoods. Second, the Metaphone algorithm is a very rough attempt at a phonetic mapping. A statistical algorithm, perhaps with a lattice output, would be better, and could help in adapting these methods to ASR output. Along these lines, we do not actually need a named entity recognizer – in fact, looking for potential matches in unstructured text is more robust, especially when the text is ASR output.

Lastly, the unsupervised method for learning edit probabilities proved quite successful. We are interested in studying other problems with independent sources of information to see if we can learn parameters in a similar unsupervised fashion.

Acknowledgments: This work was supported by the Defense Advanced Research Projects Agency (DARPA) GALE project, under Contract. No. HR0011-06-C-0023. Any opinions, findings, and conclusions or reccommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

7. References

[1] W. Cohen, P. Ravikumar, and S. Fienberg, "A comparison of string distance metrics for name-matching tasks," in *Proceedings of IIWeb Workshop*, 2003.

[2] Andrew McCallum, Kedar Bellare, and Fernando Pereira, "A conditional random field for discriminatively-trained finite-state string edit distance," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

[3] M.D. Kernighan, K.W. Church, and W.A. Gale, "A spelling correction program based on a noisy channel model," in *13th International Conference on Computational Linguistics*, 1990, pp. 205–210.

[4] E. Brill and R. Moore, "An improved error model for noisy channel spelling correction," in *ACL*, 2000, pp. 286–293.

[5] BAE, "Go/no-go formal distillation evaluation plan for gale," Tech. Rep., 2006.

[6] Ralph Grishman, David Westbrook, and Adam Meyers, "NYU's english ace 2005 system description," Tech. Rep., NYU, 2005.

[7] Jean Carletta, "Assessing agreement on classification tasks: The kappa statistic," *Computational Linguistics*, vol. 22, no. 2, pp. 249–254, 1996.

[8] Mehran Sahami and Timothy Heilman, "A web-based kernel function for measuring the similarity of short text snippets," in *WWW'06*, 2006.

[9] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 6, no. 1, pp. 391–407, 1990.

[10] David Freedman, *Statistical models: theory and practice*, pp. 174–177, cambridge university press, 2005.

[11] Lawrence Philips, "The double-metaphone search algorithm," *C/C++ User's Journal*, vol. 18, no. 6, 2000.

[12] Michael Levit, Dilek Hakkani-Tür, Gokhan Tür, and Daniel Gillick, "Integrating several annotation layers for statistical information distillation," in *IEEE Automatic Speech Recognition and Understanding Workshop*, 2007.