Matching Inconsistently Spelled Names in Automatic Speech Recognizer Output for Information Retrieval

Hema Raghavan and James Allan
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003, USA
{hema,allan}@cs.umass.edu

Abstract

Many proper names are spelled inconsistently in speech recognizer output, posing a problem for applications where locating mentions of named entities is critical. We model the distortion in the spelling of a name due to the speech recognizer as the effect of a noisy channel. The models follow the framework of the IBM translation models. The model is trained using a parallel text of closed caption and automatic speech recognition output. We also test a string edit distance based method. The effectiveness of these models is evaluated on a name query retrieval task. Our methods result in a 60% improvement in F1. We also demonstrate why the problem has not been critical in TREC and TDT tasks.

1 Introduction

Proper names are key to our understanding of topics in news. For example, to determine that a news story is on the 2004 elections in the United States, the words *President Bush*, *John Kerry* and *USA* are necessary features of the story. In other words, names of people, places and organizations are key entities of a news story. For many tasks, like in topic detection and tracking (TDT), the entities form an important feature for distinguishing topics from one another. For example, it is the people that distinguish stories on the 2004 election from stories on the 2000 U.S election. Names, especially rare and foreign ones are a problem for automatic speech recognition (ASR) systems as they are often out of vocabulary (OOV) i.e., they do not exist in the lexicon of the ASR system. An OOV word is replaced with the most similar word in the lexicon of the speech recognizer. Sometimes, even if a name is in the lexicon of the speech recognizer, it may have multiple spelling variants. The following is a sample ASR snippet from the TDT3 [1] corpus that demonstrates how the same entity may have different spellings even within the same snippet of ASR text.

...newspaper quotes **qaddafi** *is saying they'll turn them over but only if they're allowed ..leader moammar* **gadhafi** *says he doesn't want an international confrontation over the suspects in the..*

In this work, we aim to find methods by which to cluster or group names in ASR text. We evaluate a variety of techniques that range from a simple string-edit distance model to generative models using both intrinsic and extrinsic evaluations. We get statistically significant improvements in results for ad-hoc retrieval when the query is just the name of a person. We also explain why the problem of misspelled proper names in ASR has not been an issue in the TREC spoken document retrieval (SDR) track or in topic detection and tracking (TDT). We demonstrate how the problem would be of significance when the query is short, containing mainly names with little or no context.

[1] http:///www.ldc.upenn.edu/projects/tdt3/

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 451–458, Vancouver, October 2005.

2 Related Work

That names can be spelled differently is a problem that has been addressed by the database community in great detail. They found that the problem was rising in significance with the increasing interest in reconciling different databases. Differences in names due to spelling errors, spelling variants and transliteration errors have been dealt with by different kinds of approximate string matching techniques like Soundex, Phonix, and String Edit distance (James C. French, 1997; Zobel and Dart, 1996). The nature of the problem is identical when the domain consists of databases of documents but in order to apply techniques that were developed for names by the database community one would have to first detect names in the corpus, and then normalize them to some canonical form. This is the approach taken by Raghavan and Allan (Raghavan and Allan, 2004) who showed that normalizing names using Soundex codes resulted in a 10% improvement on the TDT3 Story Link Detection Task. They tested their method on newswire stories only. Their difficulty in applying Soundex to the ASR documents was that detecting names in ASR is too error prone for their methods to be useful (Miller et al., 2000).

Spoken document retrieval was a track at the TREC-6,7 and 8 (Voorhees and Harman, 1997; Voorhees and Harman, 1998; Voorhees and Harman, 1999) conferences. At the TREC-8 SDR track the conclusion was that ASR is not really an issue for ad hoc retrieval. However, the queries in those tracks were not centered on any entity. The TREC-8 proceedings also acknowledge that mean average precision dropped as named entity word error rate (NEWER) increased. A typical speech recognizer has a lexicon of about 60K and for this size of a lexicon, about 10% of the person names are out of vocabulary (OOV).

The problem of alternate spellings of names has also been explored by the cross lingual information retrieval community (Virga and Khudanpur, 2003; AbdulJaleel and Larkey, 2003). The problem with names in machine translated text is quite similar to the problem with names in ASR text, except that the errors caused by a speech recognizer are often phonetic confusions, which is not necessarily the case for machine translation errors. Spelling errors of names in machine translated text are typically consistent. A given word in the source language always translates to the same word in the target language for a given machine translation system. As seen earlier, ASR systems do not exhibit such consistency.

Another problem that resembles the one we are addressing in this paper is that of spelling correction. Spelling correction has been tackled in several different ways (Durham et al., 1983), in some cases with the use of contextual cues (Golding and Roth, 1999) and in some cases it has been modeled as a "noisy channel problem" (Kernighan et al., 1990). The latter approach is similar to ours because we also approach the problem of spelling variations due to speech recognizer errors as analogous to the errors caused by a noisy channel. However, spelling correction methods must rectify human errors (typographic errors and common confusions) whereas speech recognizer errors are different.

Additionally, the argument that *Jon Smith* and *John Smythe* may genuinely be different people and should not be considered to be the same entity is more of a cross-document co-reference problem. The problem we are attempting to solve in this paper is one of grouping names that "sound like" each other together, without considering the problem of cross document co-reference. For example, the name *Lewinsky* has 199 occurrences in the TDT3 corpus, and also appears as *Lewinski* (1324 times), and *Lewenskey* (171 times). Most of these occurrences refer to *Monica Lewinsky*. The aim is to group all these variants together, without taking into consideration which ones refer to the same person. We then measure the effectiveness of our methods on various retrieval tasks.

Perhaps the most similar work from the point of view of the task is work in word spotting in audio output (Amir et al., 2001). The queries are single words and the task is to locate their mention in audio. The starting point in that work is however, a phonetic transcript of the audio signal and the emphasis is not on locating names. Our starting point is automatic speech recognizer output, and we aim to locate names in particular.

3 Our Approaches

In this section we explain the techniques by which we group names together. One method uses string edit distance to group names that are variants of each

other. The other techniques are some of the possible generative models suitable to this task.

An **equivalence class** is defined as a group of names such that any two names in that class are variants of each other and such that there exist no two names from different equivalence classes that are variants of each other. An equivalence class is represented as a set of names enclosed in curly braces as {*name-1 name-2 ...*}

Four of our models are trained on a parallel text of ASR and manual transcripts (or closed caption depending on availability) in order to learn a probabilistic model of ASR errors. The parallel text consists of pairs of sentences: sentences from the ASR output and the corresponding manual transcripts. This is a common technique in machine translation for which the IBM translation models are popular methods (Brown et al., 1993).

As a convention, we use uppercase letters to denote ASR output and lowercase for manual transcriptions. Given an input of parallel text of ASR and manual transcriptions, the model learns a probabilistic dictionary. The dictionary contains pairs of closed caption and ASR words and the probability that the closed caption word is generated from a given word in ASR. Thus, the model might learn a high probability for *P(CAT|kate)*.

3.1 Overview of Methods

We generate equivalence classes of names by clustering a list of names. The algorithm draws links between pairs of words and then clusters the words into equivalence classes such that if a and b are linked and b and c are linked then *a*, *b* and c are in the same equivalence class. Links between words are generated in five different ways described below.

In the first of our methods we align manual transcripts and ASR sentences using the IBM translation model (Brown et al., 1993) to obtain a probabilistic dictionary. We give details of the translation model in section 3.2. Names are grouped such that if *P(CAT|kate)* is high (above some threshold) then there is a link between *CAT* and *kate*. This is called the *Simple Aligned* method. Some sample pairs of words obtained by this technique are shown in figure 1.

We can also ask a human to create a list of equivalence classes of names. We describe our method of obtaining such a list in section 4. This method is called the *Supervised* method.

african	AFRICA	albania	ALBANIAN
alex	ALEC	cardoso	CARDOZO
ann	ANNE	ching	CHIANG

Figure 1: Example of pairs of words obtained by Simple Aligned

Given a list of equivalence classes, pairs of names that go together can easily be generated such that for each pair, both words are obtained from the same equivalence class. In this way equivalence classes of names obtained from the Simple Aligned and Supervised methods can be used to create a list of pairs of names that form parallel text to train a character level machine translation model. We would expect this model to learn a high probability for similar sounding alphabets, e.g., a high probability for $P(C|k)$. Depending on where the training set of pairs of names for this method comes from, we get two possible systems. These are called the *Generative Unsupervised* method and *Generative Supervised* method respectively. Note that the Generative Unsupervised method is not completely unsupervised; we still need the parallel text of ASR and manual transcripts, but we don't need a human to do the added grouping of names into equivalence classes. A character level translation model helps us generalize better to unseen words.

We also grouped together names that differ by a string edit distance of one, giving a fifth system. In particular, we use the Levenshtein distance (Levenshtein, 1966), that is the number of insertions, deletions and substitutions needed to convert one string to the other. Many methods employed by the database community build on string edit distance. The method works well but has some disadvantages. Consider a user who types in a query containing a name such that the spelling, as typed by the user, never occurs in the corpus. To employ string edit distance, one would have to compare the query name against all the words in the vocabulary of the corpus to find the most similar strings. With a generative model, only the query needs to be expanded using the translation model, thereby speeding up the search process. The string edit distance model on the

other hand, is completely unsupervised and needs no training in the form of parallel text. Both methods have their advantages and disadvantages, and the use of one method over the other is situation dependent.

3.2 Details

To learn alignments, translation probabilities, etc in the first method we used work that has been done in statistical machine translation (Brown et al., 1993), where the translation process is considered to be equivalent to a corruption of the source language text to the target language text due to a *noisy channel*. We can similarly consider that an ASR system corrupts the spelling of a name as a result of a noisy channel. To obtain the closed caption word c, of an ASR word a, we want to find the string for which the probability $P(c|a)$ is highest. This is modeled as

$$P(c|a) = \frac{P(c)P(a|c)}{P(a)} \quad (1)$$

For a given name a, since $P(a)$ is constant, the problem reduces to one of maximizing $P(c)P(a|c)$. $P(c)$ is called the language model. We need to model $P(a|c)$ as opposed to directly modeling $P(c|a)$ so that our model assigns more probability to well formed English names.

Given a pair of sentences (c, a), an alignment $\mathcal{A}(c, a)$ is defined as the mapping from the words in c to the words in a. If there are l closed caption words and m ASR words, there are 2^{lm} alignments in $\mathcal{A}(c, a)$. $l \in \mathcal{A}(c, a)$ can be denoted as a series $l_1^m = l_1, l_2...l_m$ where $l_j = i$ means that a word in position j of the ASR string is aligned with a word in position i of the closed caption string. Then $P(a|c)$ is computed as follows:

$$\begin{aligned} P(a|c) &= \sum_l P(a, l|c) \\ P(a, l|c) &= P(m|c) \prod_j^m P(l_j|l_1^{j-1}, a_1^{j-1}, m, e) \\ &\quad \times P(a_j|l_1^j, a_1^{j-1}, m, c) \end{aligned} \quad (2)$$

where a_j is a word in position j of the string a, and a_1^j is the series $a_1...a_j$. The model is generative in the following way: we first choose for each word in the closed caption string the number of ASR words that will be connected to it, then we pick the identity of those ASR words and finally we pick the actual positions that these words will occupy. There are five different IBM translation models (Brown et al., 1993). Models 3 and 4 build on the above equations, and also incorporate the notion of fertility. Fertility takes into account that a given word in closed caption may be omitted by an ASR system, or one word may result in two or more, like *Iraq* $\rightarrow$ *I ROCK* (This is a true example). The models are trained using Expectation Maximization. Further details are in the original paper (Brown et al., 1993).

The IBM models have shown good performance in machine translation, and especially so within certain families of languages, for example in translating between French and English or between Sinhalese and Tamil (Brown et al., 1993; Weerasinghe, 2004). Pairs of closed caption and ASR sentences or words (as the case may be) are akin to a pair of closely related languages.

For the Generative Unsupervised and Generative Supervised methods, we use the same models, but in this case the training set consists of pairs of words obtained from the ASR and closed caption text as opposed to sentences. In other words, the place of words in the previous case is taken by characters. Modeling fertility, etc, again fits very well in this case. For example the terminal character e is often dropped in ASR, and a single o in closed caption may result in a double o in ASR or vice versa.

4 Experimental Set Up

4.1 Corpora

For experiments in this paper we used the TREC-6 and TREC-7 SDR track data (Voorhees and Harman, 1998). We also used the TDT2 and TDT3 corpora. For TREC-6 we had the ASR output provided by NIST (WER 34%). The TREC-7 corpus consists of the output of the Dragon systems speech recognizer (WER 29.5%). For the TDT sources we had the ASR output of the BBN Byblos Speech recognizer provided by the LDC. NIST provides human generated transcripts for the TREC corpora and LDC provides closed caption quality transcripts with a WER of 14.5% for the TDT corpora. There are 3943, 23282, 1819 and 2866 ASR documents in the TDT2 TDT3, TREC-6 and TREC-7 corpora respectively.

4.2 Intrinsic Evaluation

The Paice evaluation (Paice, 1996) for stemming algorithms (algorithms that reduce a word to its morphological root), attempts to compare the equivalence classes generated by our methods with human judgments.

The Paice evaluation measures the performance of a stemmer based on its understemming and overstemming indices (UI and OI respectively). UI measures the total number of missed links between words and OI measures the total number of false alarm links. A perfect stemmer would have a UI and OI value of zero.

We obtained a list of names to be grouped into equivalence classes in the following way. We did not use a named entity tagger on the corpus because named entity taggers typically have very high word error rates for ASR text (Bikel et al., 1999). Instead we ran the Unix *spell* command on the corpus and used the list of rejected words as the list of names for the annotators to group into equivalence classes. These 296 OOV words are taken to correspond to the names in the corpus. We then obtained the set of ground-truth equivalence classes by a method similar to Paice.

A group of undergraduate students was hired. The list of names was provided to each student in a text editor in alphabetical order. The purpose as explained to them was to group together names that were alternate spellings of similar sounding names together. The student was instructed to go through the list systematically, and for each word to look at the previous 10 words, as well as the following 10 words to see if there were any other variants. If there was a word or a group where the current word was likely to fit in, they were asked to *cut* the word and *paste* it into the appropriate group. In this way, groups were created such that no word could belong to more than one group. The annotators were also asked to mark the words that were indeed names. Of the 296 OOV words, 292 were found to be actual names.

4.3 Extrinsic evaluation

In addition to the Paice evaluation we propose two extrinsic or task based evaluations for our methods. In the first task, given a name as a query, we aim to find all documents that have a mention of that name or any of its variants. In order to obtain queries and relevance judgments for this task we arbitrarily chose 35 groups of names from the ground-truth set of equivalence classes. The TDT3 corpus was chosen to be the test corpus for this task. Hence we eliminated those words that had no occurrence in the TDT3 corpus from the 35 groups of names giving a total of 76 names. Each of the 76 words formed a query. For each name query we consider all documents that contain a mention of any of the names in the equivalence class of the query as relevant to that query. In this way we obtained relevance judgments for the name query task. Some sample queries are shown in figure 2. We use $F1$ (harmonic mean of the precision and recall) as a measure of performance.

Query Equivalence class
1: {*christy christie*}
2: {*christina christine*}
3: {*toney toni*}
4: {*michelle michel mitchell*}
5: {*columbia colombia colombian*}

Figure 2: Some sample query equivalence classes

Our extrinsic evaluation is spoken document retrieval. The queries on the TREC-6 and TREC-7 corpora are standard TREC spoken document retrieval track queries. For the TDT2 corpus we use one randomly chosen document from each topic as the query. This document is like a long query with plenty of entities and plenty of contextual information. For the TDT3 corpus we use the topic descriptions as provided by the LDC as the queries. The LDC topic descriptions discuss the *events* that describe a topic and the key entities and locations involved in the event. These are representative of shorter queries, rich in entities. LDC has provided relevance judgments for both the TDT2 and TDT3 corpora. Mean average precision was used as the measure of evaluation.

4.4 Implementation Details

We use GIZA++ (Och and Ney, 2003) to train the machine translation system and the ISI ReWrite Decoder (ISI, 2001) to do the actual translations. The decoder takes as input the models learned by

GIZA++ and a sentence from the foreign language. It can output the top n translations of the input sentence. The ReWrite decoder can translate using IBM Model-3 or Model-4. We found Model 3 to have lower perplexity and hence chose it for our experiments. In order to build the language model $P(c)$, we used the CMU Language Modeling toolkit [2]. All retrieval experiments were performed using the LEMUR [3] toolkit, and using the traditional vector space model. In the traditional vector space model queries and documents are represented as vectors of words. Each word in the vector is weighted using a product of term frequency and inverse document frequency. The similarity between a query and a document is measured using the cosine of the angle between the query and document vectors.

The Simple Aligned and Generative Unsupervised methods require a parallel corpus of ASR and closed caption for training. For the name query task we used TDT2, TREC-6 and TREC-7 to train these methods and TDT3 as the test corpus.

The Supervised and Generative Supervised methods require a human to provide pairs of words that are variants of each other. We filtered out those words from the human generated list of equivalence classes that occurred exclusively in the test corpus and in no other corpus. This is equivalent to asking a human to group words in the training corpus. Similarly we trained the Simple Aligned and Generative Unsupervised models using ASR and closed caption text from all other sources except those in the test set.

The models were trained similarly for the SDR experiments. The models were tested on each of the four corpora in turn, and in each case they were trained on everything but the test corpus.

5 Results

5.1 Intrinsic Experiments

Table 1 shows how the different methods perform on the intrinsic evaluation. We also show the UI and OI values for methods that use string edit distances of 2, 3, 4 and 5. Note that the Supervised method is the ground truth for this evaluation, and hence it has a UI and OI value of zero. A string edit distance of 1 has the lowest OI value, meaning there are very few false alarms. Higher string edit distances have lower UI values, with an increase in OI. We will interpret the UI and OI values again after observing performance on the retrieval tasks, so as to interpret the impact of missed links and false alarm links for retrieval.

Method	UI	OI
Simple Aligned	0.236	0.004
Supervised	0	0
Gen Sup	0.393	0.023
Gen Uns	0.351	0.003
Str. Ed. (1)	0.229	0.000
Str. Ed. (2)	0.083	0.003
Str. Ed. (3)	0.039	0.001
Str. Ed. (4)	0.031	0.124
Str. Ed. (5)	0.023	0.336

Table 1: Understemming and Overstemming indices for each of the methods (lower is better)

5.2 Name Query Retrieval experiments

The results of our experiments on the name query task are given in table 2. We report both Macro and Micro averaged (averaged over the equivalence classes of the queries) F1 measures. They do not differ much since the equivalence classes have almost the same number (2-3) of names.

From table 2, all methods improve the baseline F1 score significantly (statistical significance measured using a two tailed t-test with a confidence of 95%). In general, the Simple Aligned, Generative Unsupervised and string edit distance methods are the best performing for this task. The string edit distance improves the baseline by over 60%. The Supervised method is also not as good as the other four of our methods as it does not generalize well to names that occur exclusively in the test set.

String edit distance performs very well on certain equivalence classes of names. For example, on the equivalence class {*Seigal, Segal, Siegal, Siegel*} the precision and recall are 100% each since all of the words in the equivalence class differ from each other by a string edit distance of one. In the case of the equivalence class {*Lewenskey Lewinski Lewinsky*}, the term *Lewenskey* has a string edit distance of 2 (greater than one) from the other two members,

[2] http://mi.eng.cam.ac.uk/prc14/toolkit_documentation.html
[3] http://www.cs.cmu.edu/lemur

Method	Micro avg Recall	Micro avg Precision	Micro F1	Macro avg Recall	Macro avg Precision	Macro F1
Baseline	0.401	1	0.573	0.400	1	0.571
Simple Aligned	0.632	0.933	**0.754**	0.608	0.925	**0.734**
Sup	0.477	0.961	**0.638**	0.463	0.960	**0.625**
Gen Sup	0.530	0.937	**0.677**	0.517	0.938	**0.667**
Gen Uns	0.590	0.921	**0.720**	0.576	0.913	**0.706**
Str. Ed	0.752	0.867	**0.806**	0.751	0.871	**0.807**

Table 2: Results on the Name Query Retrieval task

Lewinsky and *Lewinski*. The equivalence class of {*John Jon Joan*} has very low precision and recall. This is because both *John* and *Jon* differ by a string edit distance of one from so many other names in the corpus, such as *Jong*, resulting in lowered precision.

The Simple Aligned method fails on names it has not seen in the training set. However, for cases like {*Greensborough Greensboro*} the link between these two names is detected using the simple aligned method and by no other. The generative methods can detect variations in spelling due to similar sounding alphabets. For example it can detect the link between *Sydney* and *Sidney*. The generative models were also able to learn that c and k are substitutable for each other. Therefore these models could detect the links between the words in the equivalence class {*Katherine Kathryn Catherine*}.

The Simple Aligned model performs well on the extrinsic evaluations although it has a high OI value. The intrinsic evaluations use judgments by humans. The Simple Aligned method would conflate *Kofi* and *Copy* into one class if that was a genuine ASR error and the alignment was correct, but these two words would not be conflated into the same equivalence class by our annotators and would actually count as a false alarm on the intrinsic evaluations. Therefore, although the OI is high for the Simple Aligned Method, on closer examination we found that some of the false alarms were actually representative of ASR errors.

5.3 Spoken Document Retrieval

We now move on to discuss results on the SDR task. For TDT3 we got statistically significant improvements (an improvement in mean average precision from 0.715 to 0.757) over the baseline using string edit distance. On the remaining corpora we got little or no improvement by our methods. We proceed to explain why this is the case for each of the corpora.

The TREC-7 corpus has only 5 queries with a mention of a name resulting in hardly any gains overall. Similar was the case for TREC-6. Again in the case of the TDT2 corpus, since we used entire documents as stories, there are enough words in the query that a few recognition errors can be tolerated and therefore traditional retrieval is good for the task. There is evidence from previous TREC tracks (Voorhees and Harman, 1999) that shorter queries result in a decrease in retrieval performance and hence we see some improvements for TDT3. Besides, the TDT3 queries were rich in names.

We wanted to check how our methods performed on outputs of different ASR systems. Spoken document retrieval on the TREC-7 data with the output of Dragon systems, which has a word error rate of 29.5%, results in an improvement of 6% using the Simple Aligned method. The NIST-B2 system with a higher WER (46.6%) has an improvement in Mean Average Precision of 6.5%. Similarly with the CUHTK (WER 35.6%) and NIST-B1 (WER 33.8%) and Sheffield (WER 24.6 %) systems we obtained improvements of 1.6%, 0.39% and 0.05% respectively using the Simple Aligned method. Thus, with increasing WER, the named entity word error rate increases significantly, and therefore the benefits of our method are more apparent in such situations.

6 Discussion and Conclusions

We showed (both intrinsically and extrinsically) that string edit distance is an effective technique for locating name variants. We also developed a set of generative models and showed that they are almost

as effective at name finding and document retrieval, but are probably more efficient than string edit distance. The generative models need to be trained on parallel text and therefore require human effort for training the models. The advantage of one method over the other is dependent on the size of the corpus and the availability of resources.

The problem has not been of significance in previous TREC tasks or in TDT, because we have always escaped the problem of misspelled names by virtue of the nature of those tasks. In the TREC tasks very few queries are centered on an entity. In all the TDT tasks, one is usually required to compare entire stories with each other. A story is long enough that there are enough words that are in the vocabulary (just like a very long query) or that are correctly recognized, that the ASR errors do not really matter. Therefore, the TDT tasks also do not suffer as a result of these ASR errors.

We can improve and apply our methods to other domains like Switchboard data (Godfrey et al., 1992). Our methods also generalize well across languages since there are no language specific techniques employed.

7 Acknowledgements

This work was supported in part by the Center for Intelligent Information Retrieval and in part by SPAWARSYSCEN-SD grant number N66001-02-1-8903. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

Nasreen AbdulJaleel and Leah S. Larkey. 2003. Statistical transliteration for english-arabic cross language information retrieval. In *Proceedings of the 12th CIKM conference*, pages 139–146. ACM Press.

Arnon Amir, Alon Efrat, and Savitha Srinivasan. 2001. Advances in phonetic word spotting. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 580–582, New York, NY, USA. ACM Press.

Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231.

P. F. Brown, Steven A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Lingustics*, 19(2):263–311.

Ivor Durham, David A. Lamb, and James B. Saxe. 1983. Spelling correction in user interfaces. *Commun. ACM*, 26(10):764–773.

J. Godfrey, E. Holiman, and J. McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proceedings of the International Conference on Acoustics, Speech and Signa Processing pp. I-517-520, 1992*, pages 517–520.

Andrew R. Golding and Dan Roth. 1999. A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130.

2001. ISI rewrite decoder, http://www.isi.edu/licensed-sw/rewrite-decoder/.

Allison L. Powell James C. French. 1997. Applications of approximate word matching in information retrieval. In *Proceedings of the Sixth CIKM Conference*.

Mark D. Kernighan, Kenneth W. Church, , and William A. Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of COLING-90*, pages 205–210.

V. I. Levenshtein. 1966. Binary codes capable of correcing deletions,insertions and reversals. *Phs. Dokl.*, 6:707–710.

David Miller, Richard Schwartz, Ralph Weischedel, and Rebecca Stone. 2000. Named entity extraction from broadcast news.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Chris D. Paice. 1996. Method for evaluation of stemming algorithms based on error counting. *JASIS*, 47(8):632–649.

Hema Raghavan and James Allan. 2004. Using soundex codes for indexing names in asr documents. In *Proceedings of the HLT NAACL Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval*.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-language applications. In *Proceedings of the 26th ACM SIGIR conference*, pages 365–366. ACM Press.

E. M. Voorhees and D. K. Harman, editors. 1997. *The Sixth Text REtrieval Conference (TREC 6)*. NIST.

E. M. Voorhees and D. K. Harman, editors. 1998. *The Seventh Text REtrieval Conference (TREC 7)*. NIST.

E. M. Voorhees and D. K. Harman, editors. 1999. *The Eighth Text REtrieval Conference (TREC 8)*. NIST.

Ruvan Weerasinghe. 2004. A statistical machine translation approach to Sinhala Tamil language translation. In *SCALLA 2004*.

Justin Zobel and Philip W. Dart. 1996. Phonetic string matching: Lessons from information retrieval. In *Proceedings of the 19th ACM SIGIR Conference,(Special Issue of the SIGIR Forum)*, pages 166–172.