Proper names and their Spelling Variations in Automatic Speech Recognition output

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

ABSTRACT

Names, particularly foreign names often have ambiguous spellings in English. In Automatic Speech Recognized (ASR) documents this ambiguity is more pronounced because a speech recognizer usually chooses similar sounding names or words for those it does not find in its lexicon. The result then is a large number of different spellings for the same name, even within one document.

In this paper we propose several methods of normalizing names in an ASR corpus -i.e., grouping together names such as *Arafat, Araafat* etc which are spelling variations of the same name. Our methods range from a simple String Edit Distance model to more complex generative models that model the corruption in the spelling of names as the effect of a noisy channel. We evaluate our methods using the Paice methodology which was developed for stemming algorithms. We also demonstrate the usefulness of our methods on two tasks - a new task which attempts to find all documents containing all variants of a given name, and the traditional spoken document retrieval task. We get significant improvements on the first task. We also illustrate with several examples the nature of ASR errors and give reasons why ASR errors have not surfaced as a significant problem in the TREC Spoken Document Retrieval task and on the TDT tasks.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Experimentation, Evaluation

Keywords

Spoken Document Retrieval, Automatic Speech Recognition, Spelling variants of names

1. MOTIVATION

Proper names are often key to our understanding of the information conveyed by a document. This is particularly the case when the domain is news. For example, a document with several mentions of *George W. Bush*, *Dick Cheney*, *Baghdad* and *Saddam Hussein*, gives us a good sense of what the contents of the document may be. In comparison, other regular English words like *death*, *scud* and *missiles*, may be good indicators of more general topics like *war*, but may not give us any indication of the exact event being discussed. Linking stories that discuss the same event, like the Attack on Iraq is very useful for news filtering systems.

The name *Gaddhafi* is a good example of a name with multiple spelling variations including Qaddhafi (New York Review of Books) Qaddafi (New Republic), Gaddafi (Time), Kaddafi (Newsweek), Khadafy (Maclean's), Qadhafi (U.S. News & World Report), Qadaffi (Business Week), and Gadaffi (World Press Review) [3]. In print media (or on the web) a single source is usually consistent in its spelling of a given entity. In ASR, the problem is more pronounced as names are often Out of Vocabulary, that is they do not exist in the lexicon of the speech recognizer. The ASR system usually then outputs a similar sounding word or sequence of words in place of this Out of Vocabulary word. For a 60K word lexicon, around 10% of the names are out of vocabulary [15]. Some errors may simply be a result of recognition errors. The source level consistency is also lost as is shown in the following example, which is an ASR output of a single news segment on CNN, taken from the TDT3 corpus [6]

19981018 (1130-1200) CNN HEADLINE NEWS
...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..

A person looking for all mentions of *Gaddafi* may never find a document that spells the name as *Ghaddafi* using typical search techniques. For an average information retrieval user, this might not matter as he might find sufficient relevant documents with the spelling as *Gaddafi*. But to an intelligence official, for example recall is probably very significant. Another example is that of automatically transcribing the U.S Supreme Court proceedings (http://-www.oyez.org) and searching through them. Names of people will certainly be important in that scenario, and it is easy to imagine why a user would want to search for names in such an archive.

We discuss prior work in section 2 and illustrate how our methods differ from spelling correction and other similar problems with names in other domains. We define our problem more concretely in section 3 and explain our different approaches to solve it in section 4. The intrinsic and extrinsic evaluations are outlined in section

Submission for CIKM 2004. Please do not cite or distribute this draft paper.

5 and the data and parameters of our experiments are explained in section 6. In section 7 we describe our results where we show that String Edit distance and some of our generative models show significant improvements in retrieving documents pertaining to a given entity. We also discuss how and why the problems associated with names have not cropped up significantly in previous research in TDT and the TREC spoken document retrieval tracks, and argue that it is because of the nature of those tasks that we have been escaping the problem of ambiguity in the spellings of names, and that in the long run the problem cannot be ignored. We conclude and outline directions for future work in sections 8 and 9.

2. PAST 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 as we were increasingly interested in reconciling different sources of 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 [10, 25]. The nature of the problem is identical when the domain consists of databases of documents. But solving it poses the additional question that what is it that you want to normalize? Would one aim at normalizing all words in all documents? In traditional information retrieval clustering regular English words into groups based on their morphological root is common and the technique is called stemming[12, 14]. However stemming algorithms leave names unchanged or treat them as ordinary words. The rules that would apply to conflate names would be different from stemming rules. In order to apply the techniques that were developed for names by the database community one would have to detect names in a corpus, and normalize them to some canonical form. Raghavan and Allan [19] showed that normalizing names using Soundex codes resulted in a 10% improvement on the TDT3 Story Link Detection Task, where the test set consisted of only newswire stories. Their bottleneck in applying this to the entire Story Link Detection Task was that detecting names in ASR is an errorful process.

Spoken Document Retrieval was a track at the TREC-6,7 and 8 [20, 21, 22] conferences. At the TREC-8 SDR track the conclusion was that ASR is not really an issue for traditional IR. However, the queries in those tracks were not really centered on any entity. The TREC-8 proceedings also acknowledge that Mean Average Precision dropped as Named Entity Word Error Rate (NE-WER) increased. Miller et al [15] give a breakdown of the OOV rates for different entities for different lexicon sizes. 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). When a word is OOV the speech recognizer substitutes the word for a similar sounding word or sequence of words.

The problem has been explored by the Cross Lingual IR community where the challenge is cross language transliteration variations [23, 7]. Their problem is quite similar to ours except that the domain is different and the errors caused by a Speech recognizer are typically different from those of a Machine Translation system.

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 [8], in some cases with the use of contextual cues [9] and in some cases it has been modeled as a "noisy channel problem" [11]. The last is interesting because we also approach the problem of correcting spelling variations due to speech recognizer errors as a noisy channel. However, the nature of the two problems are different; we elaborate on this distinction in the next section where we define the problem more concretely.

3. PROBLEM DEFINITION

There has been a significant amount of work on spelling correction in the past. In this section, we define our problem more precisely and also explain why it differs from spelling correction.

Differences in spellings of words can be of two types: spelling errors or spelling variations. We define spelling errors to be those kind of errors where a given word X which has one and only one correct spelling is spelled in some other form, say X'. The job of a spelling correction algorithm is to correct X' to X. Spelling errors may be due to a confusion on the part of the human or may even be a result of typographic errors which cause letters to be added, deleted or interchanged. For example, if a user typed *acheive* when he intended to type *achieve*, that would be a spelling error. A spelling variant on the other hand , is when a word may or may not have a single correct spelling and there are many different ways in which it can be spelled. For example, if two people spelled the name *Smith* differently, one person as *Smith* and the other as *Smythe*, that would qualify as a spelling variant. These latter kind of spelling variations are more predominant in ASR documents. In other words we are trying to group names that sound like each other together. However, we do experiment with models that learn from spelling errors, to see if any knowledge of those types of errors can be leveraged to the benefit of our models.

Additionally, the argument that *Jon Smith* and *John Smythe* are probably really different people and should not be grouped together is more of a cross-document coreference problem. The problem we are attempting to solve in this paper is one of grouping names that "sound like" each other together, without the use of contextual cues like those used in cross document coreference. As an example we cite the name Lewinsky which 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 entity. We then measure the effectiveness of our methods on various tasks.

4. APPROACH

In this section we explain the techniques by which we group names together. The first technique 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. We also explain why generative models may be more efficient in certain scenarios.

The simplest way to group words that sound like each other together is to use a standard clustering algorithm with String Edit distance as a distance metric. The Edit Distance between two strings is the number of insertions, deletions and substitutions it would take to convert one string to another. The simplest form of String edit distance is the Levenshtein distance where each such operation has a cost of one. We used single link clustering for clustering words together using a distance threshold of 1.

Many methods employed by the database community build on String Edit distance. The method has some disadvantages. Consider a user who types in a query containing a name such that the spelling as typed by the user has no occurrences 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. In a news filtering domain where new stories are continuously being indexed this would mean a re-clustering of the vocabulary every time a new story is indexed, with a recalculation of all corpus statistics. With a generative model the query word or document just needs to be expanded, thereby speeding up

the search process.

4.1 Definitions

Equivalence class: 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. We represent an equivalence class as a set of words enclosed in curly braces as {*name-1 name-2 ...*}

Root form: Given an equivalence class, one word from the class is chosen at random as the root form and mentions of all other words in that class in the corpus are normalized to the root form.

Basis set: The set of equivalence classes which are used to normalize the corpus.

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.

4.2 Algorithm

The diagram in Figure 1 gives an outline of the different methods we use to cluster names into different equivalence classes. In all of our methods we cluster words which are names using a clustering algorithm. For this we need to be able to determine whether any two pairs of words are linked or not. We generate pairs of linked words in many different ways. Our starting point in each case is the parallel text of ASR and closed caption. A list of names is created by running a named entity tagger on the closed caption text, and creating a names dictionary. Typically named entity extraction tools run better on grammatically sound text as opposed to ASR. Hence we chose to create the names dictionary using the closed caption text. At several points we refer to the terms – *alignments, probabilistic dictionary* etc. These terms stem from the statistical machine translation literature, which we explain briefly and discuss why it suits our task. First we give an overview of the methods of obtaining pairs of words that go together.

In the first and most simple method (lightly shaded in figure 1) we train a statistical machine translation model using ASR and closed caption text as parallel text. The output of the training process creates a probabilistic dictionary consisting of word pairs the probability that the first word is a translation of the second. We extract word pairs with a high enough (above an empirically determined threshold) translation probability, and such that at least one word exists in the names dictionary. The result is a list of pairs of names that are variants of each other. Essentially, by this simple method we are exploiting the fact that a name like *Lewinsky* in closed caption, which represents the human spelling may appear as *Lewinski, Lowinski* etc, sufficient times in the corpus and our method will be able to detect these variants in the process of aligning the ASR and closed caption text and thus create an equivalence class {*Lewinsky, Lewinski, Lowinsky*}. This list of pairs of words is called **List-1** in the figure.

The second method of generating pairs of words that are linked to each other (shaded dark in figure 1) is to ask a human to create a list of pairs of names that are spelling variations of each other. We call this **List-2**. The problem with the first and second methods is that they do not generalize well. In other words, unless a name occurred in closed caption or in ASR, we do not have an equivalence class for it. The amount of closed caption we have is limited, and since new names constantly appear in news, we wanted a method that generalized well.

The third method is generative (the unshaded area in the figure 1). It takes a list of pairs of name variants, treats one name as a translation of the other and in this way learns an alignment of characters, and a character to character probabilistic dictionary. For example it can learn that c is replaceable or a translation of k and vice-versa with high probability. In addition, it can learn from the alignments that ei may be interchangeable for ie and vice-versa. We can obtain the list of words that we use to train the generative model from List-1 or List-2. We also have a third list of common spelling errors like *achieve* and *achieve*[2] that we use to train a generative model. We call this list List-3 . Depending on whether List-1, List-2 or List-3 is used to train GIZA++ we have Methods 3, 4 and 5. In Method-6 we train GIZA++ using a combination of List-1, List-2 and List-3.

To learn alignments, translation probabilities, etc in the first method we used work that has already been done in statistical machine translation [17], where the translation process is considered a result of a *noisy channel*. We can consider that an ASR system corrupts the spelling of a name as a result of a noisy channel. To obtain the closed caption word e, of an ASR word f we want to find the string for which the probability $P(e|f)$ is greatest. This is modeled as

$$P(e|f) = \frac{P(e)P(f|e)}{P(f)} \tag{1}$$

For a given name f since $P(f)$ is constant, the problem reduces to one of maximizing $P(e)P(f|e)$. $P(e)$ is called the language model. We need to model $P(f|e)$ as opposed to directly modeling $P(e|f)$ for the same reason as in Machine translation in order that our model assign more probability to well formed English names. A more detailed explanation is given in [17].

Give a pair of sentences (e, f), an alignment $\mathcal{A}(e, f)$ is defined as the mapping from the words in e to the words in f. If there are l closed caption words and m ASR words there are 2^{lm} alignments in $\mathcal{A}(e, f)$. $a \in \mathcal{A}(e, f)$ can be denoted as a series $a_1^m = a_1, a_2...a_m$ where $a_j = i$ 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 $P(f|e)$ is computed as follows:

$$\begin{aligned} P(f|e) &= \sum_a P(f, a|e) \\ P(f, a|e) &= P(m|e) \prod_j^m P(a_j|a_1^{j-1}, f_1^{j-1}, m, e) \\ &\quad \times P(f_j|a_1^j, f_1^{j-1}, m, e) \end{aligned} \tag{2}$$

Where f_j is a word in position j of the string f, and f_1^j is the series $f_1...f_j$. 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 the identity of those ASR words and then the actual positions that these words will occupy. IBM 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* . The models are trained using EM. Further details are in [17].

For our method three onwards, we again use the same models, but in this case the pairs of strings are ASR and closed caption words as opposed to sentences, and the place of words in the previous case is taken by characters. Modeling fertility, etc, again fits very well in this case too. 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.

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 Sinhala and Tamil [17, 24]. Pairs of closed caption and ASR sentences

or words (as the case may be) are akin to a pair of closely related languages.

For any task using the generative models we have to detect the names in the corpus and cluster them using one of the generative models. Traditional named entity recognizers have very high word error rates for ASR documents. Therefore, we use a simpler assumption which is that all non-dictionary words in the corpus are names. We run the corpus through the Unix spell command and use the list of words returned as a result of that command to be the list of words that need to be clustered. We could potentially apply our method to all words in the vocabulary, but that would be a burden on computation. Further, we would be swamped with equivalence classes which are groups of morphologically related words, which is beyond our scope. The aim is to conflate variants of names

Detecting names in ASR text being error prone we approximate the list of names to be the Out of Vocabulary words in the corpus. To see if our assumption was a good one, we ran the Unix *spell* command on the names in the annotated set that is,the names in the conflation classes used in Method-2. Of 296 names, 292 were rejected by the Unix spell command. The names that were not rejected were *cypress*, *sherry*, *henry* and *wally*. Names like *Jim*, *John* etc were also rejected by the spell command. Hence it is a reasonable approximation that the list of OOV words contains all the names.

In summary, we have 7 different methods of grouping names, depending on how the pairs of names that go together were generated.

- **Method-1: (Simple Aligned)** Closed caption and ASR text are aligned using GIZA++. The name pairs are extracted from the probabilistic dictionary. The names are grouped such that if A and B are paired, and B and C are paired, then A, B and C are put into the same equivalence class. The groupings obtained in this way form the basis set.

- **Method-2: (Supervised or Sup)** A human generated list of equivalence classes of names forms the basis set.

- **Method-3: (Generative Unsupervised or Gen Uns)** Equivalence classes are obtained by training GIZA++ using list-1 (pairs of names obtained by aligning CCAP and ASR as in Method-1). We then use the generative model thus obtained to cluster OOV words in the corpus in the following way. If A is generated by B with high probability (above a threshold), then there exists a link between A and B.

- **Method-4: (Generative Supervised or Gen sup)** Equivalence classes are obtained by training GIZA++ using a human generated list of pairs of words (Method-2) which are variants of each other, and then using the generative model thus obtained to cluster OOV words as in Method-3. The clustered OOV words form the basis set.

- **Method-5: (Generative Spelling Correction or Gen Sp. Corr.)** Equivalence classes of names are obtained by training GIZA++ using pairs of words where the second word is a spelling correction of the first, and then using the generative model thus obtained to cluster OOV words as in Method-3. Again, this cluster forms the basis set.

- **Method-6: (Generative Combined or Gen Comb)** GIZA++ is trained using a combined list of pairs of words used to train the translation models in methods 3,4 and 5. The generative model is then used to cluster OOV words. This cluster forms the basis set.

- **Method-7: (String Edit Distance or StrEd)** We also grouped together names that differ by a string edit distance of one. This is a simple baseline. If we make use of a higher String Edit distance of n we denote it by StrED-n. Otherwise, the default is a value of 1.

5. EVALUATION

We evaluate our methods for grouping names in two ways. The intrinsic evaluation, first outlined by Paice[18] for stemmers, attempts to compare the groupings of names as generated by each of our methods with a gold standard which is determined by human judgments.

5.1 Intrinsic (Paice) evaluation

The Paice evaluation measures the performance of a stemmer based on its Understemming and Overstemming Indices. The Understemming Index (UI) aims to measure how many groups of words that are actually related are not conflated to the same root form by a stemmer (i.e., the misses). The Overstemming Index (OI) on the other hand measures how many words are wrongly conflated to the same root (i.e., the false alarms). If we look at the purpose of our methods as trying to build a stemmer for names, the intrinsic evaluation measures for stemmers applies to this work. We now look at the Understemming and Overstemming indices of our various methods in order to understand the performances on the extrinsic evaluation.

The gold standard (truth set) was obtained by a method is similar to Paice[18] of obtaining relevance judgments. A group of undergraduate students was hired to do the truth judgments. The entire vocabulary of all the corpora (see section 6) was provided to the student in a text editor in alphabetical order. The purpose as explained to them was to group together words that were *alternates* of each other together. *Alternates* encompassed morphological variants, typographic errors and words that *sounded like* each other. The student was instructed to go through the list systematically, and for each word 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 student was also asked to identify groups that referred to names of people, places etc. We extracted only those groups which were marked as names for our evaluation.

The Paice evaluation measures the over-stemming and understemming indices as follows. A pair of words are considered to be linked if they belong to the same equivalence class. The evaluation assumes that we have a true set of equivalence classes. For each such equivalence class i with n_i words, there are $n_i * (n_i - 1)/2$ pairwise links. This number, which is called the Global Desired Merge Total (GDMT) is the total number of pairwise links across all classes. The Global Unachieved Merge Total (GUMT), is the number of pairwise links that the algorithm missed. The Global Wrongly Merged Total (GWMT) is the total number of word pairs that were detected as linked by our algorithm but did not exist in the truth set. The overstemming and under-stemming indices are therefore computed as:

$$\begin{aligned} \text{Under-stemming Index (UI)} &= GUMT/GDMT \\ \text{Over-stemming Index (OI)} &= GWMT/GDMT \end{aligned}$$

We illustrate this by using stem classes (not names) in the truth judgments and obtain UI and OI values for stemmers of the English Language. The Krovetz stemmer has UI and OI values of

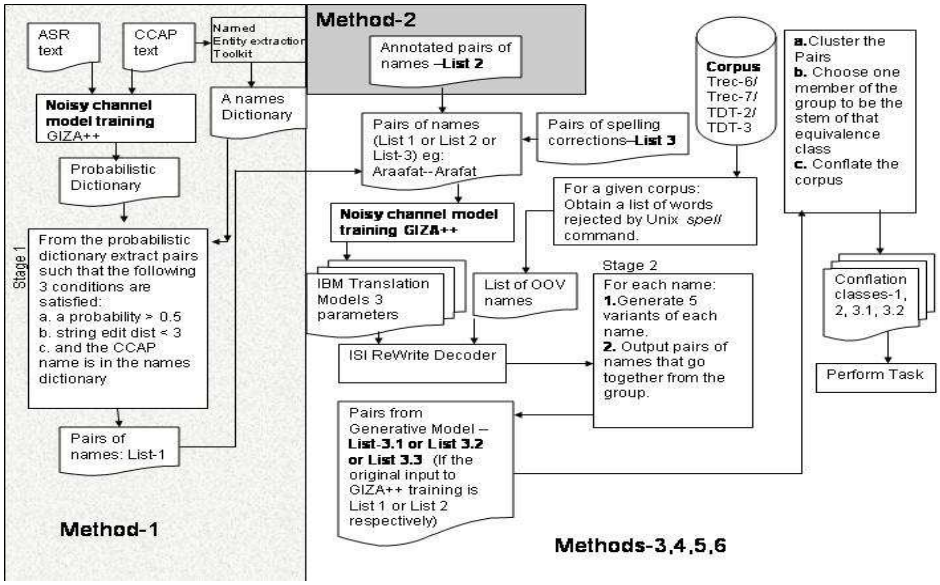

Figure 1: Architecture of System

0.5265 and 0 respectively. The Porter stemmer has UI and OI values of 0.37317 and 0 respectively. We know from the literature that a Krovetz stemmer is more conservative [12] than the porter stemmer and these results indicate exactly that. The understemming and overstemming indices indicate the strength of the stemmer, and give us a sense of the kinds of errors that may be caused by a particular stemmer. A perfect stemmer would have a UI and OI value of 0. However, in reality the trade off is usually between a high miss rate and a high false alarm rate. We know that different stemmers perform differently on different corpora and for different tasks different stemmers may be suitable (for example we know that for Arabic a light stemmer is more suitable [13]). The Paice measures allow us to quantify the performance of a stemmer, to see where it falls on the *light* to *heavy* spectrum. Thereafter, one can determine what kind of stemmer is more suited to a given task.

5.2 Extrinsic evaluations

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 of any of its variants. The truth judgments for the intrinsic evaluation were used to find documents that were relevant to a query. A document was marked as relevant to a query if it contained a mention of the query word or any other word in the equivalence class of that query word, where the equivalence classes were as defined by the truth judgments. More details on the queries and corpora are in section 6. In order to measure the performance on this task, we used set based precision and recall metrics. In other words, for each query, we measure precision as the fraction of documents of the retrieved set that are relevant and recall as the fraction of the relevant set that were retrieved. We can then average these Precision and Recall numbers that we obtain for each query, and accordingly obtain F1 measures.The F1 measure is the harmonic mean of the Precision and Recall. The average of the precision and recall can be either a Micro average or a Macro average. A micro average is an average over all queries and a Macro average is an average of the averages of each class. The micro averaged scores tend to be dominated towards the most common categories whereas Macro average scores may be skewed towards outliers. We measure both in this paper.

We evaluate our methods using some standard evaluations from Information Retrieval- ad-hoc retrieval and known item retrieval. For ad-hoc retrieval we have 3 kinds of queries: standard TREC queries, queries rich in entities with some contextual information, and thirdly queries rich in entities with lots of contextual information

For both the retrieval experiments we used a traditional information retrieval system, with a vector space model and TF*IDF weighting, with a list of 500 stopwords for stopword removal and the Krovetz stemmer for stemming [12], as a baseline. We do not expect that this baseline system will perform well. In fact, on the name query task one does not expect that to retrieve documents other than those containing the query word. Nevertheless it provides a measure of how much we stand to gain by our methods as compared to traditional methods on these tasks.

6. CORPUS, TOOLS ETC

Figure 2 shows the different corpora used in our experiments split by source and by time period. For the TDT sources we had the ASR output of the BBN Byblos Speech recognizer as given to us by LDC. For the TREC-7 data set we had the ASR output of several different systems. In experiments in this paper we used the two NIST baselines which have word error rates of about 34% and 47% respectively, CUHTK (WER-25%), Sheffield (WER 36%) and Dragon (WER 29.5%). Unless mentioned otherwise, the corpus used for TREC-7 experiments is the output of the Dragon systems speech recognizer. For TREC-6 we had only the ASR outpur provided by NIST. For the TREC corpora we have human generated transcripts and for the TDT corpora we have closed caption quality transcripts with a WER of 14.5%. The TDT3 corpus has 23282 ASR documents, and the TDT2, TREC-6 and TREC-7 corpora have 3943, 1819 and 2866 ASR documents respectively.

To train the probabilistic dictionary, alignment probabilities etc of the noisy channel model or the machine translation system we use GIZA++ [16]. In order to do the actual translations we use the ISI ReWrite Decoder [4]. The decoder takes as input the models as

Source	TDT2	TDT3	TREC-6 SDR	TREC-7 SDR
VOA	Jan-Jun 98	Oct-Dec 98	X	X
PRI	Jan-Jun 98	Oct-Dec 98	May-Jun 96	Oct 97
ABC	Jan-Jun 98	Oct-Dec 98	May-Jun 96	Jan 98
CNN	Jan-Jun 98	Oct-Dec 98	May-Jul 96	Jun 97 - Jan
NBC	X	Oct-Dec 98	X	X
MSNBC	X	Oct-Dec 98	X	X
CSPAN	X	X	Jun 96	Feb 98

Figure 2: The different corpora used in our experiments

african	AFRICA	bosnia	BOZNIA
albania	ALBANIAN	brack	BRECK
albanian	ALBANIANS	cac	CACK
alex	ALEC	calloway	CALLAWAY
america	AMERICAN	cardoso	CARDOZO
ann	ANNE	ching	CHIANG
baseball	BASEBALLS	christine	CHRISTINA
beshloss	BESCHLOSS		

Figure 3: Example of words in List-1 obtained by Simple Aligned

learned by GIZA++ and a sentence from the foreign language and can output the top n translations of the input sentence. The ReWrite decoder can translate using IBM Model-3 or Model-4. Hence, we restrict ourselves to choosing between Model 3 and Model 4. In order to build the language model $P(e)$, we used the CMU Language Modeling toolkit[1]. For our information retrieval experiments we used the LEMUR toolkit [5].

In order to obtain queries and relevance judgments for the first task, namely, the name query retrieval task, we arbitrarily chose 35 groups of names. The TDT3 corpus was chosen to be the test corpus for this task. Hence from the 35 groups of names we eliminated those words that had no occurrences in the TDT3 corpus. This gave us 35 groups with a total of 76 words. 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 name as relevant to that query. In this way we obtained relevance judgments for the name query task.

The Simple Aligned and Generative Unsupervised methods require a parallel corpus of ASR and closed caption for training. For the name query task we use TDT2, TREC-6 and TREC-7 to train these methods. The Supervised and Generative Supervised methods require a human to provide pairs of words that are variants of each other. We picked 150 groups of words at random to input to these two methods. We filtered out those words from the truth set that occurred exclusively in the TDT3 corpus and in no other corpus. This is therefore equivalent to asking a human to group words in a training corpus.

For the second type of extrinsic evaluation (Spoken Document Retrieval) we trained the different models similarly. For testing on a given corpus using the Supervised and Generative Supervised methods we filtered out those words from the truth set that occurred exclusively in that corpus and in no other 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.

Two of our four corpora for the extrinsic evaluations are the TREC-6 and TREC-7 spoken document retrieval track queries and corpora. The remaining two corpora are the TDT2 and TDT3 corpora. TREC-6 and TREC-7 queries are standard TREC queries. The TREC-6 task is known item retrieval and TREC-7 is ad-hoc retrieval. For the TDT2 corpus we use one randomly chosen document from each topic as the query. These represent long queries with lots 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.

All retrieval experiments were performed using the LEMUR toolkit, and using the traditional vector space model with TF-IDF weighting for ranked retrieval. Mean Average Precision was used as a measure of retrieval effectiveness for the evaluation of Spoken Document Retrieval on the TREC-7, TDT2 and TDT3 corpora. On the TREC-6 corpus and queries we used the evaluation used by TREC that year. The task that year was known item retrieval and the evaluation metric was the percentage of queries for which a relevant document was found at rank one. Most queries in that track had exactly one relevant document.

6.1 Parameter Settings

At stage-1 (see figure 1), we extract pairs of words from the probabilistic dictionary by choosing pairs of words with a probability of translation above an empirically derived threshold of 0.5. However, for some rare words, where the alignment makes a mistake, some of the pairs with a high probability of translation are also not acceptable. For example, on aligning the closed caption and ASR on the TDT3 corpus, we get some erroneous pairs with a high probability of translation, for example $P(THAT|Steinbrener) = 1$. Hence, we filter out those word pairs from the dictionary that have a string edit distance of greater than 3 between them. Both these thresholds were determined from the training set.

The top 15 word pairs obtained by aligning the TDT3 corpus and sorting the list alphabetically are given in figure 3. The first column is the closed caption word and the second is an ASR word. Note that the spurious terms like *baseball* in the above list are due to errors by the named entity tagger in the named entity tagging of the closed caption text, while creating the names dictionary. We did not correct those errors manually.

We used the default parameters of GIZA++ to train the Statistical Machine Translation models. An exhaustive list of the parameters is provided in the documentation [16]. Of all the possible smoothing methods that the CMU Language Modeling Toolkit [1] provides, we chose Witten-Bell smoothing as it consistently gave a lower perplexity score on a development test set, for all of our generative models.

In stage-2 we choose to expand a name word by generating the top n translations of it. With a small n the Under stemming Index is high and with a large n the Over stemming Index is high. We empirically chose $n = 5$.

7. RESULTS

7.1 Perplexity

The following table gives the Perplexity scores on a test set of 100 words for IBM models 3 and 4. Not surprisingly the translation model trained on spelling errors has higher perplexity, the nature of spelling variations in names being different from the kinds of variations caused by a speech recognizer. Of Models 3 and 4, 3 shows

lower perplexity. Therefore we use Model-3 to generate spelling variations.

	Model3	Model4
Gen Unsup	3.61314	3.2826
Gen Sup	3.10678	28.4017
Gen Sp. Corr.	8.7577	9.9129
Gen Comb	2.59981	2.91117

A few example ASR and CCAP words and the probabilities assigned by the different models are given below:

	$P(afrika\|africa)$	$P(berkely\|berkeley)$	$P(cardoza\|cardoso)$
Gen. Unsup	0.00134	0.005249	0.00022
Gen. Sup	0.00073	0.00111	0.00021
Gen. Sp. Corr.	7.02E-09	0.00562	1.57E-08
Gen. Comb	0.00202	0.00679	0.00052

7.2 Name Query Retrieval experiments

The single name retrieval experiments were performed using TREC-7, TDT2 and TREC-6 as the training corpora, and TDT3 as the test corpus. The results of our experiments on the single name query task are given in figure 4. We report both Macro and Micro averaged F1 measures (They do not differ much because in each equivalence class of query words there were mostly 2 words and a few classes had 3 words).

From the table in Figure 4, all improvements in F1 as compared to the baseline are statistically significant except the Gen Sp. Corr model. In general, the Simple aligned, Generative Unsupervised and StrED methods are the best performing for this task.

Our 76 query words for this task belonged to 35 equivalence classes. Figure 5 gives a breakdown of the F1 measures for each equivalence class for each of the methods. String edit distance performs very well on certain equivalence classes of names for example: on the equivalence class containing *Seigal* it gets a precision and recall of a 100%. The equivalence class is {*Seigal, Segal, Siegal, Siegel*} (Query class 22) and 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*} (Query class 27), the term *Lewenskey* has a string edit distance of 2 (greater than one) from the other two members *Lewinsky* and *Lewinski*. This results in lower recall and therefore a lower F1 by StrED. The equivalence class of {*John Jon Joan*} (Query class 19) 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*, and therefore this results in a lowered precision. Similarly in the case of query class {*Chiang Ching*} (Query class 11), *Chiang* and *Chang* differ from *Ching* by an edit distance of 1, resulting in a lowered precision. Whether the word *Ching* deserves to be in the same equivalence class as *Chiang* and *Chang* is debatable and is a matter of human judgment.

The Simple Aligned method relies on having seen the name in closed caption or in ASR. Therefore, for a class like {*Christina Christine*} (Query class 2) this method performs poorly. Then again, a pair like {*Greensborough Greensboro*} (Query class 24) is detected using the simple aligned method and by no other. The generative methods are able to track certain kinds of variations in spelling due to similar sounding alphabets. For example, the substitution of i with y like in the {*Sydney Sidney* } (Query class 29) equivalence class case and {*Silvia Sylvia*} (query class 23) equivalence class case. Most of the generative models have also learned that c and k are substitutable for each other as in the equivalence class {*Katherine Kathryn Catherine*} (Query class 7). Note that the generative model trained from spelling correction does well on the {*Yeltsin, Yelsin*} (query class 8) example and poorly on equivalence classes which are groups of similar sounding words like in the case of *Catherine* and *Katherine*.

Method	TREC-7	TDT3
0 (Baseline)	0.3587	0.7156
1 (Simple Aligned)	0.3646	0.7144
2 (Sup)	0.3587	0.7116
3 (Gen Sup)	0.3588	0.7146
4 (Gen Uns)	0.3659	0.7229
5 (Gen Comb)	0.3587	0.7074
6 (Gen Sp. Corr.)	0.3588	0.7186
7 (StrED)	0.3587	**0.757**

Figure 6: SDR results

7.3 Spoken Document Retrieval

We now move on to discuss results on the spoken document retrieval task. The average precision numbers for each of the methods on the different corpora are given in figure 6. The method based on clustering names using a string edit distance of 1 is better than the TF-IDF baseline, and the difference in Mean Average Precision is statistically significant on the TDT3 corpus. We experimented with higher string edit distances only to see significant decreases in Mean Average Precision.

When we look at a query-by-query breakdown of our methods most queries do not change. Query 30042 shows reasonable improvement with our methods. The topic of this query is the *Panam Lockerbie trial* and *Kofi Annan* is a key entity. The ASR versions of his names are quite different from the original. Kofi often gets corrupted by the ASR system. For example, it appears case in one case as *COPING ANAND* (document VOA19981002.1700.1937), and once as *COKIE ANAND* (VOA19981005.1700.1890). Thus query 30042 benefits by some of our methods particularly the Generative supervised method which puts *ANAN*, *ANNAN* and *ANAND* into the same equivalence class. The same goes for the String Edit distance method which groups *ANAN* and *ANNAN* into the same class. Although the name *ANN* is also a part of that same equivalence class, and there are false alarms as a result of that error, there is still an improvement in MAP of about 1.5% for this query.

On the TREC-7 Spoken Document retrieval track we see little or no improvement by our methods, except by the Simple Aligned method which causes a 6% increase in MAP and by the Unsupervised Generative method which causes a 2% increase in MAP. In the TREC-7 corpus exactly 5 queries had any mention of an OOV word. The OOV words in this query set (as determined by Unix spell command are) are *andor, canadian, chinese, chung, clinton, cuba, huang, kong montserrat.* One would expect words like *Montserrat* or *Huang* to benefit by our methods. But that did not happen. There was not a single mention of *Montserrat* in the TREC-7 ASR corpus. This word appears in Query 59 of the TREC-7 SDR track, *What data is available on volcanic activity on the island of Montserrat?*. Clearly the words *Montserrat* and *volcanic* are keywords in this query. If the word *Montserrat* had appeared as a word in the corpus then it would have been output as an OOV word, and then at least one of the methods would have attempted to cluster it with some other similar sounding words. Now we explore what the ASR system did to the different mentions of *Montserrat*. There were 6 documents relevant to the query. In one case *Montserrat* was identified as *MONTHS THE ROT* (document eo970823.srt) by the ASR system, in one case as *MONTH AROUND THE*(document

Method	Micro average Recall	Micro Average Precision	Micro F1	Macro Average Recall	Macro Average Precision	Macro F1
0 (Baseline)	0.4015	1	0.5730	0.4001	1	**0.5716**
1 (Simple Aligned)	0.6328	0.93310	**0.7542**	0.6084	0.9259	**0.7343**
2 (Sup)	0.4778	0.9613	**0.6384**	0.4637	0.9600	**0.6254**
3 (Gen Sup)	0.53022	0.9377	**0.6774**	0.5179	0.9382	**0.6676**
4 (Gen Uns)	0.5902	0.9211	**0.7200**	0.5762	0.9132	**0.7066**
5 (Gen Comb)	0.4585	0.9778	**0.6242**	0.4431	0.9803	**0.6104**
6 (Gen Sp Corr)	0.4412	0.9788	0.6082	0.4307	0.9373	0.5952
7 (StrED)	0.7525	0.8674	**0.8060**	0.7516	0.8716	**0.80724**

Figure 4: Results on the Name Query Retrieval task

Query Equivalence class	Baseline	Simpl Aligned	Sup	Gen Uns	Gen Sup	Gen Sp Corr	Gen Comb	StrED
1: {*christy christie* }	0.6692	0.6692	0.8654	0.7744	0.8174	0.6692	0.6692	0.5941
2: { *christina christine* }	0.6667	0.3102	0.9620	0.8531	0.9043	0.6667	0.6667	0.6847
3: { *toney toni*}	0.6667	0.6667	0.6667	0.6667	0.6667	0.6667	0.6667	0.3734
4: { *michelle michel mitchell*}	0.4205	0.4205	0.5391	0.4638	0.4986	0.4205	0.4205	0.8723
5: {*columbia colombia colombian*}	0.6332	0.9189	0.6332	0.7029	0.6662	0.6332	0.6332	0.4877
6: {*darryl daryl*}	0.6667	1.0000	0.6667	0.6667	0.6667	0.6667	0.6667	1.0000
7: { *katherine kathryn catherine*}	0.3268	0.7078	0.5911	0.7223	0.6502	0.2428	0.5000	0.7947
8: { *yeltsin yelsin* }	0.6968	0.6667	0.6667	0.6690	0.6679	0.7298	0.6667	0.9973
9: {*paula paul* }	0.5210	0.5210	0.5210	0.6996	0.5972	0.5210	0.5210	0.8046
10: {*kathy cathy* }	0.7750	0.6667	0.4375	0.7155	0.5430	0.6667	0.9254	0.9714
11: {*chiang ching* }	0.6667	1.0000	0.6667	0.5161	0.5818	0.6667	0.6667	0.1429
12: {*elliott elliot* }	0.4158	0.6667	0.4158	0.4314	0.4235	0.4158	0.4158	0.6667
13: {*rendell randell randall* }	0.5000	0.6667	0.6721	0.7958	0.7287	0.5000	0.5000	0.6667
14: {*ferrell farrell* }	0.7143	0.9091	0.6667	0.8293	0.7391	0.6667	0.7692	0.9091
15: {*finn flynn*}	0.5455	0.6667	0.6667	0.5333	0.5926	0.5455	0.5455	0.0250
16: {*roberta robert*}	0.6316	0.6316	0.6316	0.8520	0.7254	0.6316	0.6316	0.8556
17: {*gardiner gardner* }	0.6667	0.6667	0.6667	0.6667	0.6667	0.6667	0.6667	0.9600
18: {*jacarta jakarta* }	0.6667	1.0000	0.6667	0.6667	0.6667	0.6667	0.6667	1.0000
19: {*jon joan john* }	0.3387	0.0341	0.3390	0.3390	0.3390	0.3387	0.3387	0.3403
20: {*gennady genady* }	0.6667	0.6667	0.6667	0.6667	0.6667	0.6667	0.6667	1.0000
21: {*james jamie* }	0.6667	0.6494	0.6667	0.6667	0.6667	0.6667	0.6667	0.6667
22: {*segal siegel seigal siegal*}	0.6667	0.6667	0.6667	0.6667	0.6667	0.6667	0.6667	1.0000
23: {*silvia sylvia* }	0.6667	1.0000	0.6667	1.0000	0.8000	0.6667	0.6667	0.8571
24: {*greensboro greensborough* }	0.5455	0.6667	0.5455	0.5455	0.5455	0.5455	0.5455	0.7648
25: {*lawrence laurence* }	0.6667	0.6667	0.6667	0.9941	0.7981	0.6667	0.6667	0.9971
26: {*leona leone* }	0.6667	0.6250	0.6667	0.8506	0.7475	0.6667	0.6667	0.5000
27: { *lewenskey lewinski lewinsky* }	0.5000	0.5703	0.6585	0.7046	0.6808	0.5000	0.5000	0.9485
28: {*calloway callaway* }	0.6667	1.0000	0.6667	0.9941	0.7981	0.6667	0.6667	0.9942
29: {*sidney sydney*}	0.6667	0.9884	0.6667	0.9884	0.7963	0.6667	0.6667	0.8419
30: {*holbrook holbrooke* }	0.4225	0.6457	0.4225	0.6400	0.5090	0.4225	0.4225	0.4045
31: {*louie louis*}	0.6667	0.6667	0.6667	0.5286	0.5896	0.6667	0.6667	0.9854
32: {*lindsey linsey lindsay*}	0.5253	0.6574	0.3502	0.4485	0.3933	0.5253	0.5253	0.9908
33: {*lucile lucille* }	0.6667	1.0000	0.6667	0.7000	0.6829	0.6667	0.6667	0.9542
34: {*lynn lynne*}	0.6344	0.6344	0.6344	0.6344	0.6344	0.6344	0.6344	0.9542
35: {*macarthur mcarthur* }	0.6667	1.0000	0.6667	0.6667	0.6667	0.6667	0.6667	0.8571

Figure 5: Query by Query breakdown of the performance of our methods on the name query retrieval task

eh971027.srt). Other mentions were *MONSTER RIDE* (document eo970824.11) and *ONCE ROCK* (document eo970825.25). Therefore the word *volcanic* was the only keyword in the query that aided retrieval and that word was rarely misrecognized. We had a similar problem with the keyword *Huang* which never appeared in the ASR corpus but was a part of a query. In both these cases query expansion using the generative models might have helped. This we leave for future work.

For the TREC-6 and TDT2 queries and corpora there was no improvement using our methods. The TREC-6 SDR task was known item retrieval and the measure of performance was the percentage of queries for which a relevant document was found at rank one. All our systems including the baseline TF*IDF system achieved a retrieval rate of about 74% on the ASR corpus.. This is comparable to the best performing system that year. When we examined the queries we found the following words to be Out of Vocabulary– *Goldfinger, Unabomber, Valujet, everglades, healthcare, mammograms*. Most queries have only one relevant document, and often the above mentioned words are misrecognized in that one relevant document. For example, In the case of the query containing *Unabomber*- Query 18, and the query containing *Valujet*- Query 47, both words are misrecognized in the single relevant document and hence our methods are of no use. Some of these queries would have been hard for an information retrieval system anyways. For example, *Query 8: What is the name of a possible real model for the well-known fictional spy hero of Goldfinger and other novels?*, where the relevant document *j960617*, does not mention any of the words in the query.

Hence there was not much value to our method for two reasons, firstly the low number of questions that used Out of Vocabulary words, and secondly by definition, the task required high precision at the top of the ranked list and as long as there is a single document containing the spellings as they appear in the query, the performance is good.

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 reasonably good for the task. There has been evidence from previous TREC tracks which also support this claim [22] – that as queries get shorter, there is decrease in retrieval performance.

7.4 Intrinsic Experiments

For this set of experiments we filtered out from the true set of equivalence classes those names that did not occur in the Simple Aligned list of equivalence classes. If we did not do that the simple aligned method achieves a very high UI score. This gave us 245 equivalence classes with an average of 2.24 words per equivalence class and a total of 390 links between pairs of words that needed to be detected.

Method	UI	OI
1 (Simple Aligned)	0.23697	0.0042
2 (Sup)	0	0
3 (Gen Sup)	0.39323	0.02339
4 (Gen Uns)	0.35156	0.00391
5 (Gen Comb)	0.36198	0.00305
6 (Gen Sp. Corr)	0.67708	0.001
7 (StrED-1)	0.22917	0.00027
8 (StrED-2)	0.08333	0.0036
9 (StrED-3)	0.03906	0.001
10 (StrED-4)	0.03125	0.12448
11 (StrED-5)	0.02343	0.33671

StrED-1 has the lowest OI value. Although StrED-2 has a very low UI, its OI is much higher and therefore it performs badly on our extrinsic evaluations. Of our other models the Simple Aligned model performs well on the extrinsic evaluations although it has a high OI value. This is because the equivalence classes as obtained by the Simple Aligned method are pretty good representations of ASR confusions. However we are comparing all our methods with human confusions. For example the Simple Aligned method would conflate *Kofi* and *Copy* into one class if that was a genuine ASR error and the alignment was correct. This is not representative of human confusions 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 the false alarms were actually representative of ASR errors. For example, the previous example where Montserrat was mapped to MONTHS more than once, was clearly quite different from a human confusion. The Simple Aligned method has a UI of 0.23, which indicates that about 77% of the 390 links were detected by the Simple Aligned method. But the high OI indicates that a lot of the links as obtained by the Simple Aligned method do not feature as human confusions. Therefore, if one assumes that the Simple Aligned method is fairly accurate, the high OI indicates that a lot of ASR confusions are very different from human ones. This is probably a good explanation of why the Simple Aligned and Generative Unsupervised methods perform well on these tasks.

7.5 Other experiments

We wanted to check how our methods performed on outputs of different ASR systems. The experiments on the TREC-7 data in the previous section used the output of Dragon systems, which has a word error rate of 29.5%. The NIST-B2 system has a high WER (46.6%). For this system the improvements in Mean Average Precision using the Simple Aligned method is 6.5%. In this case too the Generative Unsupervised method performs well resulting in a 1.2% increase of Mean Average Precision as compared to the baseline.

Similarly with the CUHTK Sheff (WER 35.6%) and NIST-B1 (WER 33.8%) and (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. Now one may ask that why would one use a recognizer with a WER higher than that of the best performing system. It is important to remind ourselves that WER is not only a function of the recognition algorithm being used, but is also extremely situation dependent. For example, background noise, such as street noise or music can result in an increase of Word Error Rate.

System	NIST-B2	Drag
WER	46.6	29.5
NE-WER	77.1	40.6
MAP (Kstem)	0.2676	0.3587
MAP (Simple Al)	0.285	0.3646
MAP (Sup.)	0.2676	0.3587
MAP (Gen Sup)	0.2676	0.3588
MAP (Gen Uns)	0.2681	0.3659
MAP (Gen Spell Corr.)	0.2676	0.3587
MAP (Gen Comb)	0.2676	0.3588
MAP (Gen StrED-1)	0.2676	0.3587

8. DISCUSSION AND CONCLUSIONS

We showed that string edit distance is an effective technique for locating name variants, both intrinsically and extrinsically. However, we argued earlier that such an approach is too computationally expensive to be useful in large scale applications. We developed a

set of generative models and showed that they are almost as effective at name finding, document retrieval, and TDT tasks, but are substantially more efficient. In any setting where it is important to find variants of names across (or within) documents, the generative approaches appear to be a preferred choice.

The problem has not been of significance in previous TREC tasks or in TDT, because by virtue of the nature of those tasks we have always escaped the problem of misspelled names. In the TREC tasks few queries are centered on an entity. In Story Link Detection and other 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.

9. FUTURE WORK

We can improve and apply our methods to other domains like Switchboard data, or on the court proceedings. The Simple Aligned method uses the probabilistic dictionary learned by the statistical MT process to cluster the words. We added some heuristics to determine what pairs of words are linked from this dictionary. There are several other, more formal approaches we can adopt, for example, we can bootstrap the alignments, or re-align the corpus using the probabilistic dictionary to obtain a better probabilistic dictionary which we can incorporate in a formal framework.

Our methods also generalize well across languages. There are no language specific techniques employed in any of our methods. As long as a sufficient amount of data is available in the form of closed caption text and ASR output , our methods should work well. Our techniques would also apply in the case of Spoken Query retrieval, where the search is on a cross modal (print and ASR) corpus, where we would apply a phoneme to grapheme approach and then use our methods on the grapheme.

10. ACKNOWLEDGEMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by SPAWARSYSCEN-SD grant number N66001-02-1-8 903 . Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

11. REFERENCES

[1] The cmu language modelling toolkit, http://mi.eng.cam.ac.uk/ prc14/toolkit_documentation.html.

[2] Common spelling errors, http://www.actwin.com/rwmack/spelling.htm.

[3] How are you supposed to spell muammar gaddafi, http://www.straightdope.com/classics/a2_264b.html.

[4] Isi rewrite decoder, http://www.isi.edu/licensed-sw/rewrite-decoder/.

[5] The lemur toolkit, http://www.cs.cmu.edu/lemur.

[6] Tdt3, http://www.ldc.upenn.edu/projects/tdt3/.

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

[8] I. Durham, D. A. Lamb, and J. B. Saxe. Spelling correction in user interfaces. *Commun. ACM*, 26(10):764–773, 1983.

[9] A. R. Golding and D. Roth. A winnow-based approach to context-sensitive spelling correction. *Mach. Learn.*, 34(1-3):107–130, 1999.

[10] A. L. P. James C. French. Applications of approximate word matching in information retrieval. In *Proceedings of the Sixth CIKM Conference*, 1997.

[11] M. D. Kernighan, K. W. Church, , and W. A. Gale. A spelling correction program based on a noisy channel model. In *Proceedings of COLING-90*, pages 205–210, 1990.

[12] R. Krovetz. Viewing morphology as an inference process. In *Proceedings of the 16th SIGIR conference*, pages 191–202. ACM Press, 1993.

[13] L. S. Larkey, L. Ballesteros, and M. E. Connell. Improving stemming for arabic information retrieval: light stemming and co-occurrence analysis. In *Proceedings of the 25th SIGIR conference*, pages 275–282, 2002.

[14] P. M.F. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[15] D. Miller, R. Schwartz, R. Weischedel, and R. Stone. Named entity extraction from broadcast news, 2000.

[16] F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

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

[18] C. D. Paice. Method for evaluation of stemming algorithms based on error counting. *J. Am. Soc. Inf. Sci.*, 47(8):632–649, 1996.

[19] H. Raghavan and J. Allan. Using soundex codes for indexing names in asr documents. In *Proceedings of the HLT NAACL Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval*, 2004.

[20] In *The Sixth Text REtrieval Conference (TREC 0)*. NIST, 1997. NIST Special Publication 500-240.

[21] In *The Seventh Text REtrieval Conference (TREC 7)*. NIST, 1998. NIST Special Publication 500-242.

[22] In *The Eighth Text REtrieval Conference (TREC 8)*. NIST, 1999. NIST Special Publication 500-246.

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

[24] R. Weerasinghe. A statistical machine translation approach to sinhala tamil language translation. In *SCALLA 2004*.

[25] J. Zobel and P. W. Dart. Phonetic string matching: Lessons from information retrieval. In H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proceedings of the 19th ACM SIGIR Conference,(Special Issue of the SIGIR Forum)*, pages 166–172, 1996.