# Computing Word Similarity and Identifying Cognates with Pair Hidden Markov Models

**Wesley Mackay** and **Grzegorz Kondrak**
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
{wesleym,kondrak}@cs.ualberta.ca

## Abstract

We present a system for computing similarity between pairs of words. Our system is based on Pair Hidden Markov Models, a variation on Hidden Markov Models that has been used successfully for the alignment of biological sequences. The parameters of the model are automatically learned from training data that consists of word pairs known to be similar. Our tests focus on the identification of cognates — words of common origin in related languages. The results show that our system outperforms previously proposed techniques.

## 1 Introduction

The computation of surface similarity between pairs of words is an important task in many areas of natural language processing. In historical linguistics phonetic similarity is one of the clues for identifying *cognates*, that is, words that share a common origin (Oakes, 2000). In statistical machine translation, cognates are helpful in inducing translation lexicons (Koehn and Knight, 2001; Mann and Yarowsky, 2001), sentence alignment (Melamed, 1999), and word alignment (Tiedemann, 2003). In dialectology, similarity is used for estimating distance between dialects (Nerbonne, 2003). Other applications include cross-lingual information retrieval (Pirkola et al., 2003), detection of confusable drug names (Kondrak and Dorr, 2004), and lexicography (Brew and McKelvie, 1996).

Depending on the context, strong word similarity may indicate either that words share a common origin (*cognates*), a common meaning (*synonyms*), or are related in some way (e.g. *spelling variants*). In this paper, we focus on cognates. Genetic cognates are well-suited for testing measures of word similarity because they arise by evolving from a single word in a proto-language. Unlike rather indefinite concepts like synonymy or confusability, cognation is a binary notion, which in most cases can be reliably determined.

Methods that are normally used for computing word similarity can be divided into orthographic and phonetic. The former includes string edit distance (Wagner and Fischer, 1974), longest common subsequence ratio (Melamed, 1999), and measures based on shared character $n$-grams (Brew and McKelvie, 1996). These usually employ a binary identity function on the level of character comparison. The phonetic approaches, such as Soundex (Hall and Dowling, 1980) and Editex (Zobel and Dart, 1996), attempt to take advantage of the phonetic characteristics of individual characters in order to estimate their similarity. All of the above methods are static, in the sense of having a fixed definition that leaves little room for adaptation to a specific context. In contrast, the methods proposed by Tiedemann (1999) automatically construct weighted string similarity measures on the basis of string segmentation and bitext co-occurrence statistics.

We have created a system for determining word similarity based on a Pair Hidden Markov Model. The parameters of the model are automatically learned from training data that consists of word

pairs that are known to be similar. The model is trained using the Baum-Welch algorithm (Baum et al., 1970). We examine several variants of the model, which are characterized by different training techniques, number of parameters, and word length correction method. The models are tested on a cognate recognition task across word lists representing several Indo-European languages. The experiments indicate that our system substantially outperforms the most commonly used approaches.

The paper is organized as follows. Section 2 gives a more detailed description of the problem of word similarity. Section 3 contains an introduction to Pair Hidden Markov Models, while section 4 describes their adaptation to our domain. Sections 5 and 6 report experimental set-up and results.

## 2 Word Similarity

Word similarity is, at its core, an alignment task. In order to determine similarity between two words, we look at the various alignments that can exist between them. Each component of the alignment is assigned a probability-based score by our trained model. The scores are then combined to produce the overall similarity score for any word pair, which can be used to rank the word pairs against each other. Alternatively, a discrete cut-off point can be selected in order to separate pairs that show the required similarity from the ones that do not.

Before we can align words, they must be separated into symbols. Typically, the symbols are characters in the orthographic representation, and phonemes in the phonetic representation. We also need to put some restrictions on the possible alignments between these symbols. By adopting the following two assumptions, we are able to fully exploit the simplicity and efficiency of the Pair Hidden Markov Model.

First, we assume that the basic ordering of symbols remains the same between languages. This does not mean that every symbol has a corresponding one in the other language, but instead that word transformation comes from three basic operations: *substitution*, *insertion* and *deletion*. Exceptions to this rule certainly exist (e.g. *metathesis*), but are sufficiently infrequent to make the benefits of this constraint far outweigh the costs.

Second, we assume that each symbol is aligned to at most one symbol in the other word. This assumption is aimed at reducing the number of parameters that have to be learned from limited-size training data. If there is a many-to-one correspondence that is consistent between languages, it would be beneficial to change the word representation so that the many symbols are considered as a single symbol instead. For example, a group of characters in the orthographic representation may correspond to a single phoneme if the word is written phonetically.

## 3 Pair Hidden Markov Models

Hidden Markov Models have been applied successfully to a number of problems in natural language processing, including speech recognition (Jelinek, 1999) and statistical machine translation (Och and Ney, 2000). One of the more intangible aspects of a Hidden Markov Model is the choice of the model itself. While algorithms exist to train the parameters of the model so that the model better describes its data, there is no formulaic way to create the model. We decided to adopt as a starting point a model developed in a different field of study.

Durbin et al. (1998) created a new type of Hidden Markov Model that has been used for the task of aligning biological sequences (Figure 1). Called a Pair Hidden Markov Model, it uses two output streams in parallel, each corresponding to a sequence that is being aligned.[1] The alignment model has three states that represent the basic edit operations: substitution (represented by state "M"), insertion ("Y"), and deletion ("X"). "M", the *match state*, emits an aligned pair of symbols (not necessarily identical) with one symbol on the top and the other on the bottom output stream. "X" and "Y", the *gap states*, output a symbol on only one stream against a gap on the other. Each state has its own emission probabilities representing the likelihood of producing a pairwise alignment of the type described by the state. The model has three transition parameters: $\delta$, $\varepsilon$, and $\tau$. In order to reduce the number of parameters, there is no explicit start state. Rather, the probability of starting in a given state is equal to

[1] Pair Hidden Markov Models have been used in the area of natural language processing once before: Clark (2001) applied PHMMs to the task of learning stochastic finite-state transducers for modeling morphological paradigms.

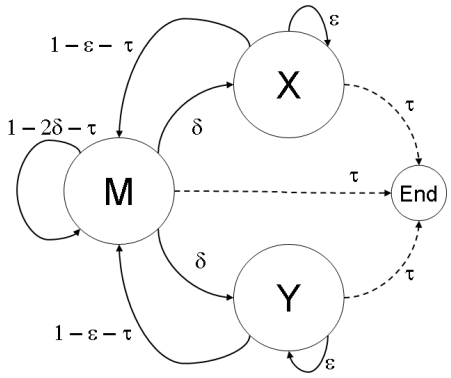


Figure 1: A Pair Hidden Markov Model for aligning biological sequences.

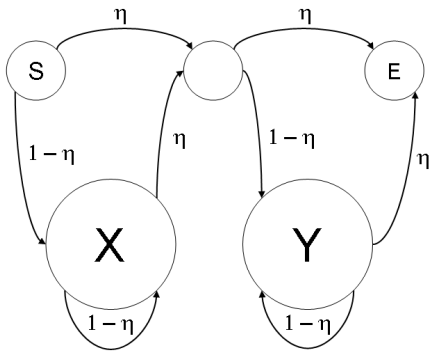


Figure 2: The random Pair Hidden Markov Model.

the probability of going from the match state to the given state.

Durbin et al. (1998) describe several different algorithms that can be used to score and rank paired biological sequences. Two of them are based on common HMM algorithms. The Viterbi algorithm uses the most probable path through the model to score the pair. The forward algorithm computes the total overall probability for a pair by summing up the probabilities of every possible alignment between the words. A third algorithm (the *log odds* algorithm) was designed to take into account how likely the pair would be to occur randomly within the two languages by considering a separately trained *random model* (Figure 2) in conjunction with the similarity model. In the random model, the sequences are assumed to have no relationship to each other, so there is no match state. The log odds algorithm calculates a score for a pair of symbols by dividing the probability of a genuine correspondence between a pair of symbols (the similarity model) by the probability of them co-occurring by chance (the random model). These individual scores are combined to produce an overall score for the pair of sequences in the same way as individual symbol probabilities are combined in other algorithms.

## 4 PHMMs for Word Similarity

Because of the differences between biological sequence analysis and computing word similarity, the bioinformatics model has to be adapted to handle the latter task. In this section, we propose a number of modifications to the original model and the corresponding algorithms. The modified model is shown in Figure 3.

First, the original model's assumption that an insertion followed by a deletion is the same as a substitution is problematic in the context of word similarity. Covington (1998) illustrates the problem with an example of Italian "due" and the Spanish "dos", both of which mean "two". While there is no doubt that the first two pairs of symbols should be aligned, there is no historical connection between the Italian "e" and the Spanish "s". In this case, a sequence of an insertion and a deletion is more appropriate than a substitution. In order to remedy this problem, we decided to a add a pair of transitions between states "X" and "Y", which is denoted by $\lambda$ in Figure 3.

The second modification involves splitting the parameter $\tau$ into two separate values: $\tau_M$ for the match state, and $\tau_{XY}$ for the gap states. The original biological model keeps the probability for the transition to the end state constant for all other states. For cognates, and other word similarity tasks, it may be that similar words are more or less likely to end in gaps or matches. The modification preserves the symmetry of the model while allowing it to capture how likely a given operation is to occur at the end of an alignment.

### 4.1 Algorithm Variations

We have investigated several algorithms for the alignment and scoring of word pairs. Apart from the standard Viterbi (abbreviated **VIT**) and forward (**FOR**) algorithms, we considered two variations of the log odds algorithm, The original log odds algorithm (**LOG**) functions much like a Viterbi algo-

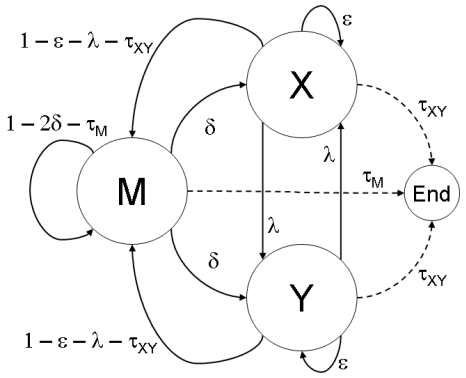


Figure 3: A Pair Hidden Markov Model for aligning words.

rithm, looking at only the most probable sequence of states. We also created another variation, forward log odds (**FLO**), which uses a forward approach instead, considering the aggregate probability of all possible paths through both models.

### 4.2 Model Variations

Apart from comparing the effectiveness of different algorithms, we are also interested in establishing the optimal structure of the underlying model. The similarity model can be broken up into three sets of parameters: the match probabilities, the gap probabilities, and the transition probabilities. Our goal is to examine the relative contribution of various components of the model, and to find out whether simplifying the model affects the overall performance of the system. Since the match probabilities constitute the core of the model, we focus on the remaining emission and transition probabilities. We also investigate the necessity of including an explicit end state in the model.

The first variation concerns the issue of gap emission probabilities. For the log odds algorithm, Durbin et al. (1998) allow the gap emission probabilities of both the similarity and random models to be equal. While this greatly simplifies the calculations and allows for the emphasis to be on matched symbols, it might be more in spirit with the word similarity task to keep the emissions of the two models separate. If we adopt such an approach, the similarity model learns the gap emission probabilities using the forward-backward algorithm, just as is done with the match probabilities, but the random model uses letter frequencies from the training data instead. A similar test of the effectiveness of trained gap parameters can be performed for the Viterbi and forward algorithms by proceeding in the opposite direction. Instead of deriving the gap probabilities from the training data (as in the original model), we can set them to uniform values after training, thus making the final scores depend primarily on matches.

The second variation removes the effect the transition parameters have on the final calculation. In the resulting model, a transition probability from any state to any state (except the end state) is constant, effectively merging "X", "Y", and "M" into a single state. One of the purposes of the separated states was to allow for affine gap penalties, which is why there are different transition parameters for going to a gap state and for staying in that state. By making the transitions constant, we are also taking away the affine gap structure. As a third variant, we try both the first and second variation combined.

The next variation concerns the effect of the end state on the final score. Unlike in the alignment of biological sequences, word alignment boundaries are known beforehand, so an end state is not strictly necessary. It is simple enough to remove the end state from our model after the training has been completed. The remaining transition probability mass is shifted to the transitions that lead to the match state.

Once the end state is removed, it is possible to reduce the number of transition parameters to a single one, by taking advantage of the symmetry between the insertion and deletion states. In the resulting model, the probability of entering a gap state is equal to $\frac{1-x}{2}$, where $x$ is the probability of a transition to the match state. Naturally, the log odds algorithms also have a separate parameter for the random model.

### 4.3 Correcting for Length

Another problem that needs to be addressed is the bias introduced by the length of the words. The principal objective of the bioinformatics model is the optimal alignment of two sequences. In our case, the alignment is a means to computing word similarity. In fact, some of the algorithms (e.g. the forward algorithm) do not yield an explicit best alignment. While the log odds algorithms have a built-in length correction, the Viterbi and the forward do not.

These algorithms continually multiply probabilities together every time they process a symbol (or a symbol pair), which means that the overall probability of an alignment strongly depends on word lengths. In order to rectify this problem, we multiply the final probability by $\frac{1}{C^n}$, where $n$ is the length of the longer word in the pair, and $C$ is a constant. The value of $C$ can be established on a held-out data set.[2]

### 4.4 Levenshtein with Learned Weights

Mann and Yarowsky (2001) investigated the induction of translation lexicons via bridge languages. Their approach starts with a dictionary between two well studied languages (e.g. English-Spanish). They then use cognate pairs to induce a *bridge* between two strongly related languages (e.g. Spanish and Italian), and from this create a smaller translation dictionary between the remaining two languages (e.g. English and Italian). They compared the performances of several different cognate similarity (or distance) measures, including one based on the Levenshtein distance, one based on the stochastic transducers of Ristad and Yianilos (1998), and a variation of a Hidden Markov Model. Somewhat surprisingly, the Hidden Markov Model falls well short of the baseline Levenshtein distance.[3]

Mann and Yarowsky (2001) developed yet another model, which outperformed all other similarity measures. In the approach, which they call "Levenshtein with learned weights", the probabilities of their stochastic transducer are transformed into substitution weights for computing Levenshtein distance: 0.5 for highly similar symbols, 0.75 for weakly similar symbols, etc. We have endeavored to emulate this approach (abbreviated **LLW**) by converting the log odds substitution scores calculated from the fully trained model into the substitution weights used by the authors.

[2]Another common method to correct for length is to take the $n^{th}$ root of the final calculation, where $n$ is the length of the longest word. However, our initial experiments indicated that this method does not perform well on the word similarity task.

[3]The HMM model of (Mann and Yarowsky, 2001) is of distinctly different design than our PHMM model. For example, the emission probabilities corresponding to the atomic edit operations sum to one for *each* alphabet symbol. In our model, the emission probabilities for different symbols are interdependent.

## 5 Experimental Setup

We evaluated our word similarity system on the task of the identification of cognates. The input consists of pairs of words that have the same meaning in distinct languages. For each pair, the system produces a score representing the likelihood that the words are cognate. Ideally, the scores for true cognate pairs should always be higher than scores assigned to unrelated pairs. For binary classification, a specific score threshold could be applied, but we defer the decision on the precision-recall trade-off to downstream applications. Instead, we order the candidate pairs by their scores, and evaluate the ranking using *11-point interpolated average precision* (Manning and Schutze, 2001).

Word similarity is not always a perfect indicator of cognation because it can also result from lexical borrowing and random chance. It is also possible that two words are cognates and yet exhibit little surface similarity. Therefore, the upper bound for average precision is likely to be substantially lower than 100%.

### 5.1 Data

Training data for our cognate recognition model comes from the Comparative Indoeuropean Data Corpus (Dyen et al., 1992). The data contains word lists of 200 basic meanings representing 95 speech varieties from the Indoeuropean family of languages. Each word is represented in an orthographic form without diacritics using the 26 letters of the Roman alphabet. All cognate pairs are also identified in the data.

The development set[4] consisted of two language pairs: Italian and Serbo-Croatian, as well as Polish and Russian. We chose these two language pairs because they represent very different levels of relatedness: 25.3% and 73.5% of the word pairs are cognates, respectively. The percentage of cognates within the data is important, as it provides a simple baseline from which to compare the success of our algorithms. If our cognate identification process

[4]Several parameters used in our experiments were determined during the development of the word similarity model. These include the random model's parameter $\eta$, the constant transition probabilities in the simplified model, and the constant $C$ for correcting the length bias in the Viterbi and forward algorithms. See (Mackay, 2004) for complete details.

were random, we would expect to get roughly these percentages for our recognition precision (on average).

The test set consisted of five 200-word lists representing English, German, French, Latin, and Albanian, compiled by Kessler (2001). The lists for these languages were removed from the training data (except Latin, which was not part of the training set), in order to keep the testing and training data as separate as possible.[5] We converted the test data to have the same orthographic representation as the training data.

### 5.2 Significance tests

We performed pairwise statistical significance tests for various model and algorithm combinations. Following the method proposed by Evert (2004), we applied Fisher's exact test to counts of word pairs that are accepted by only one of the two tested algorithms. For a given language pair, the cutoff level was set equal to the actual number of cognate pairs in the list. For example, since 118 out of 200 word pairs in the English/German list are cognate, we considered the true and false positives among the set of 118 top scoring pairs. For the overall average of a number of different language pairs, we took the union of the individual sets. For the results in Tables 1 and 2, the pooled set contained 567 out of 2000 pairs, which corresponds to the proportion of cognates in the entire test data (28.35%).

## 6 Experimental Results

In this section, we first report on the effect of model variations on the overall performance, and then we compare the best results for each algorithm.

### 6.1 Model Variations

Table 1 shows the average cognate recognition precision on the test set for a number of model variations combined with four basic algorithms, **VIT**, **FOR**, **LOG**, and **FLO**, which were introduced in Section 4.1. The first row refers to the fully trained model without changes. The remaining rows contain the results for the model variations described in Section 4.2. In all cases, the simplifications are in effect during testing only, after the full model had been trained. We also performed experiments with the model simplified prior to training but their results were consistently lower than the results presented here.

| Model Variation | Algorithm | | | |
|---|---|---|---|---|
| | VIT | FOR | LOG | FLO |
| full model | 0.630 | 0.621 | 0.656 | **0.631** |
| gaps const | 0.633 | 0.631 | *0.684* | 0.624 |
| trans const | 0.565 | 0.507 | *0.700* | 0.550 |
| both const | 0.566 | 0.531 | ***0.704*** | 0.574 |
| no end state | 0.626 | 0.620 | 0.637 | 0.601 |
| single param | **0.647** | **0.650** | *0.703* | 0.596 |

Table 1: Average cognate recognition precision for each model and algorithm combination.

With the exception of the forward log odds algorithm, the best results are obtained with simplified models. The model with only a single transition parameter performs particularly well. On the other hand, the removal of the end state invariably causes a decrease in performance with respect to the full model. If a non-essential part of the model is made constant, only the Viterbi-based log odds algorithm improves significantly; the performance of the other three algorithms either deteriorates or shows no significant difference.

Overall, the top four variations of the Viterbi-based log odds algorithm (shown in italics in Table 1) significantly outperform all other PHMM variations and algorithms. This is not entirely unexpected as **LOG** is a more complex algorithms than both **VIT** and **FOR**. It appears that the incorporation of the random model allows **LOG** to better distinguish true similarity from chance similarity. In addition, the log odds algorithms automatically normalize the results based on the lengths of the words under examination. However, from the rather disappointing performance of **FLO**, we conclude that considering all possible alignments does not help the log odds approach.

[5] The complete separation of training and testing data is difficult to achieve in this case because of the similarity of cognates across languages in the same family. For each of the removed languages, there are other closely related languages that are retained in the training set, which may exhibit similar or even identical correspondences.

| Languages | | Proportion of Cognates | Method | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | LCSR | LLW | ALINE | VIT | FOR | LOG | FLO |
| English | German | 0.590 | 0.895 | 0.917 | 0.916 | 0.932 | 0.932 | 0.930 | 0.929 |
| French | Latin | 0.560 | 0.902 | 0.893 | 0.863 | 0.916 | 0.914 | 0.934 | 0.904 |
| English | Latin | 0.290 | 0.634 | 0.713 | 0.725 | 0.789 | 0.792 | 0.803 | 0.755 |
| German | Latin | 0.290 | 0.539 | 0.647 | 0.706 | 0.673 | 0.666 | 0.730 | 0.644 |
| English | French | 0.275 | 0.673 | 0.725 | 0.615 | 0.751 | 0.757 | 0.812 | 0.725 |
| French | German | 0.245 | 0.568 | 0.591 | 0.504 | 0.556 | 0.559 | 0.734 | 0.588 |
| Albanian | Latin | 0.195 | 0.541 | 0.510 | 0.618 | 0.546 | 0.557 | 0.680 | 0.541 |
| Albanian | French | 0.165 | 0.486 | 0.444 | 0.612 | 0.505 | 0.530 | 0.653 | 0.545 |
| Albanian | German | 0.125 | 0.275 | 0.340 | 0.323 | 0.380 | 0.385 | 0.379 | 0.280 |
| Albanian | English | 0.100 | 0.245 | 0.322 | 0.277 | 0.416 | 0.406 | 0.382 | 0.403 |
| AVERAGE | | 0.2835 | **0.576** | **0.610** | **0.616** | **0.647** | **0.650** | **0.704** | **0.631** |

Table 2: Average cognate recognition precision for various models and algorithms.

### 6.2 Comparison

Table 2 contains the results of the best variants, which are shown in boldface in Table 1, along with other methods for comparison. The results are separated into individual language pairs from the test set. For the baseline method, we selected the Longest Common Subsequence Ratio (**LCSR**), a measure of orthographic word similarity often used for cognate identification (Brew and McKelvie, 1996; Melamed, 1999; Koehn and Knight, 2001). The LCSR of two words is computed by dividing the length of their longest common subsequence by the length of the longer word. **LLW** stands for "Levenshtein with learned weights", which is described in Section 4.4. We also include the results obtained by the **ALINE** word aligner (Kondrak, 2000) on phonetically-transcribed word lists.

Because of the relatively small size of the lists, the differences among results for individual language pairs are not statistically significant in many cases. However, when the average over all language pairs is considered, the Viterbi-based log odds algorithm (**LOG**) is significantly better than all other algorithms in Table 2. The differences between the remaining algorithms are not statistically significant, except that they all significantly outperform the LCSR baseline.

The fact that **LOG** is significantly better than ALINE demonstrates that given a sufficiently large training set, an HMM-based algorithm can automatically learn the notion of phonetic similarity, which is incorporated into ALINE. ALINE does not involve extensive supervised training, but it requires the words to be in a phonetic, rather than orthographic form. We conjecture that the performance of **LOG** would further improve if it could be trained on phonetically-transcribed multilingual data.

## 7 Conclusion

We created a system that learns to recognize word pairs that are similar based on some criteria provided during training, and separate such word pairs from those that do not exhibit such similarity or whose similarity exists solely by chance. The system is based on Pair Hidden Markov Models, a technique adapted from the field of bioinformatics. We tested a number of training algorithms and model variations on the task of identifying cognates. However, since it does not rely on domain-specific knowledge, our system can be applied to any task that requires computing word similarity, as long as there are examples of words that would be considered similar in a given context.

In the future, we would like to extend our system by removing the one-to-one constraint that requires alignments to consist of single symbols. It would also be interesting to test the system in other applications, such as the detection of confusable drug names or word alignment in bitexts.

## Acknowledgments

This research was funded in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Alberta Informatics Circle of Research Excellence (iCORE).

## References

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic function of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171.

Chris Brew and David McKelvie. 1996. Word-pair extraction for lexicography. In *Proceedings of the 2nd International Conference on New Methods in Language Processing*, pages 45–55.

Alexander Clark. 2001. Learning morphology with Pair Hidden Markov Models. In *Proceedings of the Student Workshop at ACL 2001*.

Michael A. Covington. 1998. Alignment of multiple languages for historical comparison. In *Proceedings of COLING-ACL'98*, pages 275–280.

Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological sequence analysis*. Cambridge University Press.

Isidore Dyen, Joseph B. Kruskal, and Paul Black. 1992. An Indoeuropean classification: A lexicostatistical experiment. *Transactions of the American Philosophical Society*, 82(5).

Stefan Evert. 2004. Significance tests for the evaluation of ranking methods. In *Proceedings of COLING 2004*, pages 945–951.

Patrick A. V. Hall and Geoff R. Dowling. 1980. Approximate string matching. *Computing Surveys*, 12(4):381–402.

Frederick Jelinek. 1999. *Statistical Methods for Speech Recognition*. The Massachusetts Institute of Technology Press.

Brett Kessler. 2001. *The Significance of Word Lists*. Stanford: CSLI Publications.

Philipp Koehn and Kevin Knight. 2001. Knowledge sources for word-level translation models. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 27–35.

Grzegorz Kondrak and Bonnie Dorr. 2004. Identification of confusable drug names: A new approach and evaluation methodology. In *Proceedings of COLING 2004*, pages 952–958.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proceedings of NAACL 2000*, pages 288–295.

Wesley Mackay. 2004. Word similarity using Pair Hidden Markov Models. Master's thesis, University of Alberta.

Gideon S. Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *Proceedings of NAACL 2001*, pages 151–158.

Christopher D. Manning and Hinrich Schutze. 2001. *Foundations of Statistical Natural Language Processing*. The MIT Press.

I. Dan Melamed. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, 25(1):107–130.

John Nerbonne. 2003. Linguistic variation and computation. In *Proceedings of EACL-03*, pages 3–10.

Michael P. Oakes. 2000. Computer estimation of vocabulary in protolanguage from word lists in four daughter languages. *Journal of Quantitative Linguistics*, 7(3):233–243.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL-2000*, pages 440–447.

Ari Pirkola, Jarmo Toivonen, Heikki Keskustalo, Kari Visala, and Kalervo Jarvelin. 2003. Fuzzy translation of cross-lingual spelling variants. In *Proceedings of SIGIR'03*, pages 345–352.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2):522–532.

Jörg Tiedemann. 1999. Automatic construction of weighted string similarity measures. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, College Park, Maryland.

Jörg Tiedemann. 2003. Combining clues for word alignment. In *Proceedings of the 10th Conference of the European Chapter of the ACL (EACL03)*.

Robert A. Wagner and Michael J. Fischer. 1974. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173.

Justin Zobel and Philip Dart. 1996. Phonetic string matching: Lessons from information retrieval. In *Proceedings of SIGIR'96*, pages 166–172.