Multiple Word Alignment with Profile Hidden Markov Models

Aditya Bhargava and Grzegorz Kondrak
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada, T6G 2E8
{abhargava,kondrak}@cs.ualberta.ca

Abstract

Profile hidden Markov models (Profile HMMs) are specific types of hidden Markov models used in biological sequence analysis. We propose the use of Profile HMMs for word-related tasks. We test their applicability to the tasks of multiple cognate alignment and cognate set matching, and find that they work well in general for both tasks. On the latter task, the Profile HMM method outperforms average and minimum edit distance. Given the success for these two tasks, we further discuss the potential applications of Profile HMMs to any task where consideration of a set of words is necessary.

1 Introduction

In linguistics, it is often necessary to align words or phonetic sequences. Covington (1996) uses alignments of cognate pairs for the historical linguistics task of comparative reconstruction and Nerbonne and Heeringa (1997) use alignments to compute relative distances between words from various Dutch dialects. Algorithms for aligning pairs of words have been proposed by Covington (1996) and Kondrak (2000). However, it is often necessary to align multiple words. Covington (1998) proposed a method to align multiple words based on a handcrafted scale of similarity between various classes of phonemes, again for the purpose of comparative reconstruction of languages.

Profile hidden Markov models (Profile HMMs) are specific types of hidden Markov models used in biological sequence analysis, where they have yielded success for the matching of given sequences to sequence families as well as to multiple sequence alignment (Durbin et al., 1998). In this paper, we show that Profile HMMs can be adapted to the task of aligning multiple words. We apply them to sets of multilingual cognates and show that they produce good alignments. We also use them for the related task of matching words to established cognate sets, useful for a situation where it is not immediately obvious to which cognate set a word should be matched. The accuracy on the latter task exceeds the accuracy of a method based on edit distance.

Profile HMMs could also potentially be used for the computation of word similarity when a word must be compared not to another word but to another set of words, taking into account properties of all constituent words. The use of Profile HMMs for multiple sequence alignment also presents applications to the acquisition of mapping dictionaries (Barzilay and Lee, 2002) and sentence-level paraphrasing (Barzilay and Lee, 2003).

This paper is organized as follows: we first describe the uses of Profile HMMs in computational biology, their structure, and then discuss their applications to word-related tasks. We then discuss our data set and describe the tasks that we test and their experimental setups and results. We conclude with a summary of the results and a brief discussion of potential future work.

2 Profile hidden Markov models

In computational biology, it is often necessary to deal with multiple sequences, including DNA and protein sequences. For such biological sequence analysis, Profile HMMs are applied to the common tasks of simultaneously aligning multiple related sequences to each other, aligning a new sequence to

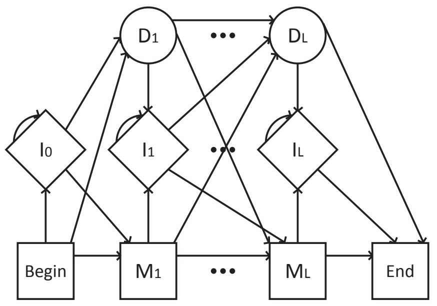

Figure 1: A prototypical Profile HMM of length *L*. M_i is the ith match state, I_i is the ith insert state, and D_i is the ith delete state. Delete states are silent and are used to indicate gaps in a sequence.

an already-aligned family of sequences, and evaluating a new sequence for membership in a family of sequences.

Profile HMMs consist of several types of states: match states, insert states, delete states, as well as a begin and end state. For each position in a Profile HMM, there is one match state, one insert state, and one delete state. A Profile HMM can thus be visualized as a series of columns, where each column represents a position in the sequence (see Figure 1). Any arbitrary sequence can then be represented as a traversal of states from column to column.

Match states form the core of the model; each match state is represented by a set of emission probabilities for each symbol in the output alphabet. These probabilities indicate the distribution of values for a given position in a sequence. Each match state can probabilistically transition to the next (i.e. next-column) match and delete states as well as the current (i.e. current-column) insert state.

Insert states represent possible values that can be inserted at a given position in a sequence (before a match emission or deletion). They are represented in the same manner as match states, with each output symbol having an associated probability. Insert states are used to account for symbols that have been inserted to a given position that might not otherwise have occurred “naturally” via a match state. Insert states can probabilistically transition to the next match and delete states as well as the current insert state (i.e. itself). Allowing insert states to transition to themselves enables the consideration of multiple-symbol inserts.

```
MMIIIM
AG...C
A-AG.C
AG.AA-
--AAAC
AG...C
```

Figure 2: A small DNA multiple alignment from (Durbin et al., 1998, p. 123).

Similarly, delete states represent symbols that have been removed from a given position. For a sequence to use a delete state for a given position indicates that a given character position in the model has no corresponding characters in the given sequence. Hence, delete states are by nature silent and thus have no emission probabilities for the output symbols. This is an important distinction from match states and insert states. Each delete state can probabilistically transition to the next match and delete states as well as the current insert state.

Figure 2 shows a small example of a set of DNA sequences. The match columns and insert columns are marked with the letters M and I respectively in the first line. Where a word has a character in a match column, it is a match state emission; when there is instead a gap, it is a delete state occurrence. Any characters in insert columns are insert state emissions, and gaps in insert columns represent simply that the particular insert state was not used for the sequence in question.

Durbin et al. (1998) describe the uses of Profile HMMs for tasks in biological sequence analysis. Firstly, a Profile HMM must be constructed. If a Profile HMM is to be constructed from a set of aligned sequences, it is necessary to designate certain columns as match columns and others as insert column. The simple heuristic that we adopt is to label those columns match states for which half or more of the sequences have a symbol present (rather than a gap). Other columns are labelled insert states. Then the probability a_{kl} of state k transitioning to state l can be estimated by counting the number of times A_{kl} that the transition is used in the alignment:

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$$

Similarly, the probability $e_k(a)$ of state k emitting symbol a is estimated by counting the number of times $E_k(a)$ that the emission is used in the alignment:

$$e_k(a) = \frac{E_k(a)}{\sum_{a'} E_k(a')}$$

There is the danger that some probabilities may be set to zero, so it is essential to add pseudocounts. The pseudocount methods that we explore are described in section 3.

If a Profile HMM is to be constructed from a set of unaligned sequences, an initial model is generated after which it can be trained to the sequences using the Baum-Welch algorithm. The length of the model must be chosen, and is usually set to the average length of the unaligned sequences. To generate the initial model, which amounts to setting the transition and emission probabilities to some initial values, the probabilities are sampled from Dirichlet distributions.

Once a Profile HMM has been constructed, it can be used to evaluate a given sequence for membership in the family. This is done via a straightforward application of the forward algorithm (to get the full probability of the given sequence) or the Viterbi algorithm (to get the alignment of the sequence to the family). For the alignment of multiple unaligned sequences, a Profile HMM is constructed and trained as described above and then each sequence can be aligned using the Viterbi algorithm.

It should also be noted that Profile HMMs are generalizations of Pair HMMs, which have been used for cognate identification and word similarity (Mackay and Kondrak, 2005) between pairs of words. Unlike Pair HMMs, Profile HMMs are position-specific; this is what allows their application to multiple sequences but also means that each Profile HMM must be trained to a given set of sequences, whereas Pair HMMs can be trained over a very large data set of pairs of words.

3 Adapting Profile HMMs to words

Using Profile HMMs for biological sequences involves defining an alphabet and working with related sequences consisting of symbols from that alphabet. One could perform tasks with cognates sets in a similar manner; cognates are, after all, related words, and words are nothing more than sequences of symbols from an alphabet. Thus Profile HMMs present potential applications to similar tasks for cognate sets. We apply Profile HMMs to the multiple alignment of cognate sets, which is done in the same manner as multiple sequence alignment for biological sequences described above. We also test Profile HMMs for determining the correct cognate set to which a word belongs when given a variety of cognate sets for the same meaning; this is done in a similar manner to the sequence membership evaluation task described above.

Although there are a number of Profile HMM packages available (e.g. HMMER), we decided to develop an implementation from scratch in order to achieve greater control over various adjustable parameters.[1] We investigated the following parameters:

Favouring match states When constructing a Profile HMM from unaligned sequences, the choice of initial model probabilities can have a significant effect on results. It may be sensible to favour match states compared to other states when constructing the initial model; since the transition probabilities are sampled from a Dirichlet distribution, the option of favouring match states assigns the largest returned probability to the transition to a match state.

Pseudocount method We implemented three pseudocount methods from (Durbin et al., 1998). In the following equations, $e_j(a)$ is the probability of state j emitting character a. c_{ja} represents the observed counts of state j emitting symbol a. A is the weight given to the pseudocounts.

Constant value A constant value AC is added to each count. This is a generalization of Laplace's rule, where $C = \frac{1}{A}$.

$$e_j(a) = \frac{c_{ja} + AC}{\sum_{a'} c_{ja'} + A}$$

Background frequency Pseudocounts are added in proportion to the background frequency q_a, which is the frequency of occurrence of character a.

$$e_j(a) = \frac{c_{ja} + Aq_a}{\sum_{a'} c_{ja'} + A}$$

[1]Our implementation is available online at `http://www.cs.ualberta.ca/~ab31/profilehmm`.

Substitution matrix (Durbin et al., 1998) Given a matrix $s(a, b)$ that gives the log-odds similarity of characters a and b, we can determine the conditional probability of a character b given character a:

$$P(b|a) = q_b e^{s(a,b)}$$

Then we define f_{ja} to be the probability derived from the counts:

$$f_{ja} = \frac{c_{ja}}{\sum_{a'} c_{ja'}}$$

Then the pseudocount values are set to:

$$\alpha_{ja} = A \sum_b f_{jb} P(a|b)$$

Finally, the pseudocount values are added to the real counts as above:

$$e_j(a) = \frac{c_{ja} + \alpha_{ja}}{\sum_{a'} c_{ja'} + \alpha_{ja'}}$$

Pseudocount weight The weight that the pseudocounts are given (A in the above equations).

Smoothing during Baum-Welch The problem has many local optima and it is therefore easy for the Baum-Welch algorithm to get stuck around one of these. In order to avoid local optima, we tested the option of adding pseudocounts during Baum-Welch (i.e. between iterations) rather than after it. This serves as a form of noise injection, effectively bumping Baum-Welch away from local optima.

4 Data for experiments

Our data come from the Comparative Indoeuropean Data Corpus (Dyen et al., 1992). The data consist of words in 95 languages in the Indoeuropean family organized into word lists corresponding to one of 200 meanings. Each word is represented in the English alphabet. Figure 3 shows a sample from the original corpus data. We manually converted the data into disjoint sets of cognate words, where each cognate set contains only one word from each language. We also removed words that were not cognate with any other words.

On average, there were 4.37 words per cognate set. The smallest cognate set had two words (since we excluded those words that were not cognate with any other words), and the largest had 84 words. There were on average 10.92 cognate sets in a meaning. The lowest number of cognate sets in a meaning was 1, and the largest number was 22.

```
a 026 DAY
...
b                      003
  026 53 Bulgarian       DEN
  026 47 Czech E         DENY
  026 45 Czech           DEN
  026 43 Lusatian L      ZEN
  026 44 Lusatian U      DZEN
  026 50 Polish          DZIEN
  026 51 Russian         DEN
  026 54 Serbocroatian   DAN
  026 42 Slovenian       DAN
  026 41 Latvian         DIENA
  026 05 Breton List     DEIZ, DE(Z)
  026 04 Welsh C         DYDD
  026 20 Spanish         DIA
  026 17 Sardinian N     DIE
  026 11 Ladin           DI
  026 08 Rumanian List   ZI
  026 09 Vlach           ZUE
  026 15 French Creole C ZU
  026 13 French          JOUR
  026 14 Walloon         DJOU
  026 10 Italian         GIORNO
...
```

Figure 3: An excerpt from the original corpus data. The first two numbers denote the meaning and the language, respectively.

5 Multiple cognate alignment

Similar to their use for multiple sequence alignment of sequences in a family, we test Profile HMMs for the task of aligning cognates. As described above, an initial model is generated. We use the aforementioned heuristic of setting the initial model length to the average length of the sequences. The transition probabilities are sampled from a uniform-parameter Dirichlet distribution, with each parameter having a value of 5.0. The insert-state emission probabilities are set to the background frequencies and the match-state emission probabilities are sampled from a Dirichlet distribution with parameters set in proportion to the background frequency. The model is

```
MIIMIIMI        MIIMIIMI
D--E--N-        D--E--NY
Z--E--N-        DZ-E--N-
DZIE--N-        D--A--N-
DI-E--NA        D--E--IZ
D--I--A-        D--Y--DD
D--I--E-        Z-----U-
Z--U--E-        Z-----I-
J--O--UR        D-----I-
DJ-O--U-        G--IORNO
```

Figure 4: The alignment generated via the Profile HMM method for some cognates. These were aligned together, but we show them in two columns to preserve space.

trained to the cognate set via the Baum-Welch algorithm, and then each word in the set is aligned to the model using the Viterbi algorithm. The words are added to the training via a summation; therefore, the order in which the words are considered has no effect, in contrast to iterative pairwise methods.

The setting of the parameter values is discussed in section 6.

5.1 Results

To evaluate Profile HMMs for multiple cognate alignment, we analyzed the alignments generated for a number of cognate sets. We found that increasing the pseudocount weight to 100 improved the quality of the alignments by effectively biasing the model towards similar characters according to the substitution matrix.

Figure 4 shows the Profile HMM alignment for a cognate set of words with the meaning “day.” As with Figure 2, the alignment’s first line is a guide label used to indicate which columns are match columns and which are insert columns; note that consecutive insert columns represent the same insert state and so are not aligned by the Profile HMM. While there were some duplicate words (i.e. words that had identical English orthographic representations but came from different languages), we do not show them here for brevity.

In this example, we see that the Profile HMM manages to identify those columns that are more highly conserved as match states. The ability to identify characters that are similar and align them correctly can be attributed to the provided substitution matrix.

Note that the characters in the insert columns should not be treated as aligned even though they represent emissions from the same insert state (this highlights the difference between match and insert states). For example, Y, A, Z, D, R, and O are all placed in a single insert column even though they cannot be traced to a single phoneme in a protoform of the cognate set. Particularly infrequent characters are more likely to be put together than separated even if they are phonetically dissimilar.

There is some difficulty, also evident from other alignments we generated, in isolating phonemes represented by pairs of characters (digraphs) as singular entities. In the given example, this means that the *dz* in *dzien* was modelled as a match state and then an insert state. This is, however, an inherent difficulty in using data represented only with the English alphabet, which could potentially be addressed if the data were instead represented in a standard phonetic notation such as IPA.

6 Cognate set matching

Evaluating alignments in a principled way is difficult because of the lack of a gold standard. To adjust for this, we also evaluate Profile HMMs for the task of matching a word to the correct cognate set from a list of cognate sets with the same meaning as the given word, similar to the evaluation of a biological sequence for membership in a family. This is realized by removing one word at a time from each word list and then using the resulting cognate sets within the meaning as possible targets. A model is generated from each possible target and a log-odds score is computed for the word using the forward algorithm. The scores are then sorted and the highest score is taken to be the cognate set to which the given word belongs. The accuracy is then the fraction of times the correct cognate set is identified.

To determine the best parameter values, we used a development set of 10 meanings (roughly 5% of the data). For the substitution matrix pseudocount method, we used a log-odds similarity matrix derived from Pair HMM training (Mackay and Kondrak, 2005). The best results were achieved with favouring of match states enabled, substitution-matrix-based pseudocount, pseudocount weight of 0.5, and pseudocounts added during Baum-Welch.

6.1 Results

We employed two baselines to generate scores between a given word and cognate set. The first baseline uses the average edit distance of the test word and the words in the given cognate set as the score of the word against the set. The second baseline is similar but uses the minimum edit distance between the test word and any word in the given cognate set as the score of the word against the entire set. For example, in the example set given in Figure 4, the average edit distance between *zen* and all other words in the set is 2.58 (including the hidden duplicate words) and the minimum edit distance is 1. All other candidate sets are similarly scored and the one with the lowest score is considered to be the correct cluster with ties broken randomly.

With the parameter settings described in the previous section, the Profile HMM method correctly identifies the corresponding cognate set with an accuracy of 93.2%, a substantial improvement over the average edit distance baseline, which obtains an accuracy of 77.0%.

Although the minimum edit distance baseline also yields an impressive accuracy of 91.0%, its score is based on a single word in the candidate set, and so would not be appropriate for cases where consideration of the entire set is necessary. Furthermore, the baseline benefits from the frequent presence of duplicate words in the cognate sets. Profile HMMs are more robust, thanks to the presence of identical or similar characters in corresponding positions.

7 Conclusions

Profile HMMs present an approach for working with sets of words. We tested their use for two cognate-related tasks. The method produced good-quality multiple cognate alignments, and we believe that they could be further improved with phonetically transcribed data. For the task of matching words to correct cognate sets, we achieved an improvement over the average edit distance and minimum edit distance baselines.

Since Profile HMM training is highly sensitive to the choice of initial model, we would like to explore more informed methods of constructing the initial model. Similarly, for building models from unaligned sequences, the addition of domain knowledge would likely prove beneficial. We also plan to investigate better pseudocount methods, as well as the possibility of using n-grams as output symbols.

By simultaneously considering an entire set of related words, Profile HMMs provide a distinct advantage over iterative pairwise methods. The success on our tasks of multiple alignment and cognate set matching suggests applicability to similar tasks involving words, such as named entity recognition across potentially multi-lingual corpora.

Acknowledgements

We thank Qing Dou for organizing the cognate sets from the original data. We are also grateful to the anonymous reviewers for their valuable comments. This research was funded in part by the Natural Sciences and Engineering Research Council of Canada.

References

Regina Barzilay and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *Proc. of EMNLP*, pages 164–171.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proc. of NAACL-HLT*, pages 16–23.

Michael A. Covington. 1996. An algorithm to align words for historical comparison. *Computational Linguistics*, 22(4):481–496.

Michael A. Covington. 1998. Alignment of multiple languages for historical comparison. In *Proc. of COLING-ACL*, pages 275–279.

Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. 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).

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

Wesley Mackay and Grzegorz Kondrak. 2005. Computing word similarity and identifying cognates with pair hidden Markov models. In *Proc. of CoNLL*, pages 40–47.

John Nerbonne and Wilbert Heeringa. 1997. Measuring dialect distance phonetically. In *Proc. of the Third Meeting of ACL SIGPHON.*