Structural alignment for finite-state syntactic processing

Brian Roark
Center for Spoken Language Understanding (CSLU)
OGI School of Science & Engineering
Oregon Health & Science University (OHSU)
20000 NW Walker Road
Beaverton, Oregon 97006
roark@cslu.ogi.edu

Abstract

In this technical report, we present some preliminary experiments on using multiple sequence alignment (MSA) techniques for inducing monolingual finite-state tagging models that capture some global sequence information. Such MSA techniques are popular in biological sequence processing, where key information about long-distance dependencies and three-dimensional structures of protein or nucleotide sequences can be captured without resorting to polynomial complexity context-free models. In the natural language processing (NLP) community, such techniques have been used very little – most notably for aligning paraphrases [4] – and not at all for monolingual syntactic processing. We discuss key issues in pursuing this approach: syntactic functional alignment; inducing multiple sequence alignments; and using such alignments in tagging. Experiments are preliminary but promising.

1 Introduction

Finite-state (Markov) modeling approaches are justifiably popular for annotation of hidden structure, such as shallow (non-hierarchical) phrases, due to algorithms for inference that have complexity linear in the length of the string (Viterbi) and very effective learning algorithms, e.g., conditional random fields. Yet for some syntactic annotation tasks, the use of context-free models can provide relatively large improvements over finite-state models, hence potentially justifying the relatively expensive inference (cubic complexity in the length of the string). For example, [10] showed a greater than one percent improvement achieved by mixing the Charniak parser [5] with a perceptron trained model replicating the [16] chunking results, yielding a new best for that chunking task. [9] showed that a 4% absolute gain was achieved in a discourse segmentation task when features derived from context-free parse trees were used, versus features derived from strictly finite-state annotations.

To the extent that some of the information available from context-free analyses can be embedded into finite-state models, perhaps some of the accuracy gain from these examples could be achieved without requiring full context-free processing. Finite-state approximations to context-free grammars have been the topic of numerous papers in the field of computational linguistics [15, 14, 13], and robust tagging models have also been used to capture some aspects of syntactic structure efficiently [17]. While strings of several hundred words do occur with enough frequency to make the complexity of context-free inference onerous, the length of biological sequences (e.g., DNA/RNA or protein sequences) are typically far longer. It is often the grammar constant that makes exact inference with high accuracy context-free models (such as those in [5] or [6]) intractable in NLP; yet even basic context-free models become too costly when sequences of several thousand symbols are the norm, as in biological sequence processing. As a result, finite-state methods form the core set of algorithms for processing even higher order structure for these biological sequences. See, e.g., [8] for a presentation of some of these approaches.

This technical report presents a preliminary investigation of the use of multiple sequence alignment (MSA) techniques for finite-state natural language syntactic processing. While pairwise alignment methods are ubiquitous in NLP, multiple sequence alignment methods have not been widely used in NLP, despite being heavily used for biological sequence processing. Barzilay and Lee used an MSA approach for learning lexical choice [3] and paraphrasing [4] models within the broad context of natural language generation systems. MSA techniques have also been used in feature exploration for discourse analysis [1]. These uses of MSA have

```
---T--C---C-G--------------C----T-G---A-TA-G---AT---G-G-----G-CTC-GCG--T-CTG--A
------G---T-G--------------G----T-A---T-AA-G---AT---G-G-----A-CCC-GCG--T-TGG--A
------G---T-G--------------G----T-A---T-AG-G---AT---G-G-----A-CCC-GCG--T-CTG--A
------G--GC-G--------------G----T-G---A-AG-G---AT---G-A-----G-CCC-GCG--G-CCT--A
------C---C-G--------------G----T-A---G-AC-G---AT---G-G-----G-GAT-GCG--T-TCC--A
---T--C---C-G--------------C----T-T---T-GA-G---AT---G-G-----C-CTC-GCG--T-CCG--A
```

Figure 1: Columns 1623-1703 out of 7683 of aligned prokaryote 16S rRNA, from aligned sequences available through `http://greengenes.lbl.gov/`

explored alignments based on lexical identity or semantic similarity. For the current work, we will align sequences based on syntactic function, derived from hierarchical syntactic parse trees, which is a departure from previous uses of alignment in NLP.

The rest of this paper is structured as follows. We first present background on multiple sequence alignments in biological sequence processing, including reference to commonly used algorithms for constructing them. This is followed by a section detailing our approach to syntactic alignment. Finally, we will present some (very) preliminary experimental results on the approaches we have outlined and their use for finite-state syntactic processing.

2 Multiple sequence alignments

In biological sequence processing, sets of evolutionarily related sequences (families) are jointly aligned in a tabular format. For example, Figure 1 shows real MSA data for bacterial RNA sequences. RNA sequences come from a 4 symbol alphabet $\{\texttt{A},\texttt{C},\texttt{T},\texttt{G}\}$. The '-' symbol represents a gap in the sequence, and symbols from different sequences (rows) that occupy the same column in the MSA are aligned.

Given an MSA, simple statistical alignment models can be built, which are typically called profile models. Position specific score matrices (PSSM) assign a cost to each of the five symbols (the alphabet plus the gap) at each column. From an existing MSA, the PSSM parameters can be estimated as the negative log probability of the symbol given the column.[1] Note that this sort of model provides no symbol insertion mechanism between columns, which may prove necessary. A *profile hidden Markov model* [8] provides such a mechanism, and can also be estimated from a given MSA, using relative frequency estimation and simple smoothing.

Optimal alignment of an input string with a PSSM or profile HMM can be achieved via a dynamic programming table with the number of columns on one side and the input string on the other side. Hence simple alignment of a string S with an MSA which has M columns takes O($M|S|$) in space and time.

Induction of an MSA from a set of un-aligned input strings has been an area of substantial research in biological sequence processing. For this paper, we consider a very simple method called *iterative pairwise alignment* [2]. The algorithm requires the calculation of all pairwise distances between the strings to be aligned. Initialize the alignment with one string from the set[2]. Then, until all strings have been aligned:

1. Choose two strings S_i and S_j such that
   - S_i is already in the alignment
   - S_j is not in the alignment
   - The distance between S_i and S_j is the minimum distance between any such pair
2. Pairwise align S_i and S_j
3. Incorporate S_j into the MSA, respecting both the existing MSA and the alignment of S_i with S_j

Iterative refinement techniques [2] take an existing MSA, remove aligned strings, and re-align them using models derived from the MSA after the string was removed, such as a PSSM. For the trials in the current paper, combinations of iterative pairwise alignment with subsequent iterative refinement were the methods used to induce the MSA. Specific details will be provided in section 4.

[1] To smooth such distributions, typically Laplace's rule (add one observation per symbol per column) is used, along with relative frequency estimation.

[2] For results in this paper, we typically used the center string, or that string with the smallest sum of distances to all strings in the set, as the initial string in the alignment.

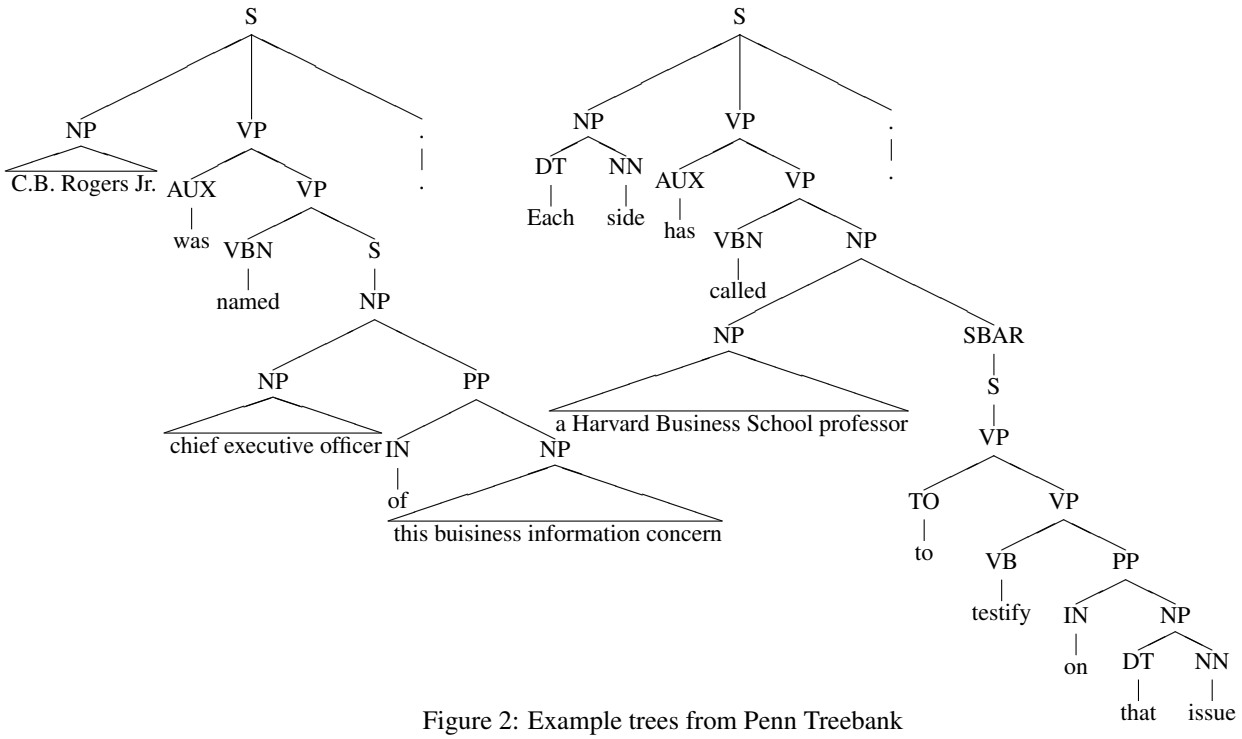

Figure 2: Example trees from Penn Treebank

3 Syntactic alignment

Our approach to syntactic alignment is based on trees from the Penn WSJ Treebank [12]. Constituents are aligned in a top-down fashion, and the head words of aligned constituents are designated to be aligned. For example, the trees in Figure 2 are first aligned at the root symbol (S), at which point the heads of these root S nodes are aligned. All subsequent alignments must then respect the fact that the heads of the S nodes are aligned.

We applied standard head percolation rules, along the lines proposed in [11], with the exception that the head of a VP is an embedded VP in preference to the AUX or MD verbs. Hence, for the trees in Figure 2, '*named*' and '*called*' are aligned at the root of the tree, as the heads of the two sentences.

After the root nodes are aligned, the children of the root node are aligned. For this example, both S nodes rewrite to {NP VP .}, hence those categories are aligned. Once the categories are aligned, their heads are aligned. Finally, their children nodes are aligned, and so on down to the leaves of the tree.

While these trees align quite nicely near the root of the tree, as the alignment proceeds, fewer perfect matches between the children occur. In order to align as much of the strings as possible, we first make a 'strict' alignment pass through the tree, aligning categories when they match perfectly; then we perform a 'loose' pass through the tree, respecting the alignments previously committed in the 'strict' pass, but permitting further alignments that did not occur in the first pass.

Informally, the algorithm is as shown in Figure 3, which also shows the alignment that results between the two strings in Figure 2 through the use of this algorithm. There are a couple of details to note. First, we pre-defined substitution costs between preterminal POS-tags, so that tags that can play a similar syntactic role (e.g., different kinds of nouns, noun modifiers, verbs, verb modifiers, punctuation, etc.) have less cost of substitution than between POS-tags of different classes.

In addition, note that this alignment approach can provide us with an alternative alignment to the full tree-based alignment by simply removing all of the internal structure of the tree: a single root node rewriting in one production to all of the POS tags. The returned alignment will essentially be the Levenshtein minimum edit distance alignment between POS-tag sequences, taking into account the pairwise substitution costs between POS-tags. Whereas the alignment based on the full tree will consistently align the heads of the string and key phrases in the tree, the alignment based on a flat tree will be uninformed by constituents or heads of the string.

Informal algorithm

1. Try to align two tree nodes τ_1 and τ_2
2. If labels of nodes differ, try to align τ_1 or τ_2 with the children of the other
3. Align children labels, **respecting already committed alignments**
4. Let m be the number of matches
   Let n_i the number of unaligned non-POS non-terminals among children of τ_i
5. If $m = 0$, try to align τ_1 or τ_2 with the children of the other
6. Assign alignments of the heads of aligned children (fixed moving forward)
7. If 'loose' alignment
   - Delete the unaligned child node that will lead to the most matches
   - Goto line 3
8. Align children nodes with aligned labels

Resulting alignment:

C.B.	Rogers	Jr.	was	named	–	–	chief	executive	officer
–	Each	side	has	called	a	Harvard	Business	School	professor

–	–	of	this	business	information	concern
to	testify	on	that	–	–	issue

Figure 3: Informal presentation of the alignment algorithm, and the resulting alignment of the strings derived from the full trees in Figure 2

4 Inducing syntactic MSA

Given an alignment method (e.g., based on full trees or flat trees as presented in Section 3), we can build an MSA for sections 2-21 of the Penn WSJ treebank, which consists of approximately 40,000 sentences (1 million words). To do this, we first require a matrix of pairwise distances for the given alignment method. We defined the distance for the trials in this paper as follows: align the two trees with the given method (full or flat); count the number of unaligned words in each string; divide by the total number of words in both strings.

Given these distances, we follow the iterative pairwise alignment algorithm presented in Section 2. However, rather than using this to align all 40 thousand strings into the MSA, we instead stopped the iterative pairwise alignment after just 4000 strings had been incorporated into the MSA, at which point we used an induced PSSM model to align the remaining strings into the MSA. We then iterated multiple times: using the PSSM to create and MSA, then using the MSA to induce the PSSM. This resulted in a reduction of columns in the MSA, by virtue of columns eventually containing only gaps across the entire corpus, at which point the columns can be removed. Table 1 shows the number of columns at each iteration, for each of the two alignment methods that we used.

At the end of this iterative process, we can use both the given MSA and the given PSSM profile model for processing new strings. New strings are aligned using POS-tags, which can either be gold tags (given by the treebank) or POS-tagger tags. In the next section, we provide results on accuracy of column assignment given POS-tags, as well as results on using these column assignments within a finite-state base phrase parser.

5 Experiments

For experimental results, we used sections 2-21 for training of all models, section 00 for stopping of the learning algorithms, and section 24 for development of the approach. All results presented here are on

Iteration:	0	1	2	3	4	5	6
Full trees:	506	457	412	398	391	382	378
Flat trees:	1867	1083	784	716	683	669	656

Table 1: Number of columns in MSA at each iteration of PSSM estimation

Anchors	NNP	NNP	,	CD	NNS	JJ	,	MD	VB	DT	NN	IN	DT	JJ	NN	NNP	CD	.
∞	5	22	67	84	97	119	123	127	132	185	209	235	257	272	280	343	359	373
100	1	13	39	45	51	61	64	65	71	97	109	121	139	151	155	179	189	197
10	0	0	0	1	3	4	4	4	5	7	11	13	15	16	17	18	18	19
5	0	0	0	0	1	2	2	2	3	4	4	4	5	6	7	8	8	9

Table 2: Columns assigned to input string of POS tags, both raw (Anchors=∞) and with different numbers of anchors for "coarse" column definitions

section 24, since this approach is still under development. Base phrase parsing (or chunking) involves non-hierarchical bracketing of constituents with only pre-terminal (POS-tag) children. Reference base phrase constituents are extracted from a treebank by removing all constituents in the tree with any child that is not a POS-tag. Evaluation consists of labeled bracketing precision and recall, combined into an F-measure accuracy score, according to standard practices for parsing evaluation.

We developed a discriminative tagging system for this work, with both training and inference modules, based on the approaches in [16, 10]. For POS-tagging, the tagset includes POS-tags, and the input string is the word sequence. For base phrase parsing, the tagset includes $\{O, \mathrm{B}_X, \mathrm{I}_X\}$ for 20 non-terminal categories X, where B_X signifies that the word is the beginning of a base-phrase labeled with X, I_X signifies that the word is inside of a base-phrase labeled with X, and O signifies that the word is outside of any base-phrase. Clearly, the tag I_X can only follow either B_X or I_X in the sequence, for the sequence to represent a valid bracketing. For the base phrase parsing task, the input string is the POS-tag sequence.

We learned models using the Perceptron algorithm, and used averaging to control for overtraining [7], which was found in [16] to be competitive with conditional likelihood optimization. Features included a subset of features explored in [16], including tritag output labels and up to 4 input word/tag sequences with the output tag (or bitag). POS-tagging performance was 96.9 on section 24, which is the same as the performance achieved by models as described in [10], a replication of the approach in [16].

Before discussing the accuracy of our PSSM in aligning new strings when given POS-tagger output, we will discuss "coarse" column representations. With 378 possible column assignments, the meaning of any individual column, in terms of syntactic function, may be too fine grained (and perhaps too sparsely observed) to be of great utility. One way to leverage the alignment is to define a small number of regions of the string by collapsing columns into a coarser representation. This can be done via what we are calling "anchor" columns, i.e., those columns that are rarely empty. Examples of such columns are those associated with the head verb of the string, the subject head noun and sentence final punctuation. By selecting those columns where gaps occur least often, we can then define regions of the MSA that occur between the selected anchors. Given k anchors, there are $k+1$ regions, leading to a total of $2k+1$ "coarse" columns.

Table 2 shows the column assignments to a particular POS-tag sequence, using the PSSM derived from full parse trees via the process detailed in Section 4. Coarse columns are also shown, when there are 100 anchors, 10 anchors or 5 anchors. Note that, when anchors are specified, odd numbers represent the anchors and even numbers represent the regions between anchors. Thus, when there are 5 anchors, the anchor associated with the main verb is column 3, and the anchor associated with the final punctuation is column 9.

Table 3 shows the accuracy of column assignments for section 24 of the WSJ treebank, using the derived PSSM models from the full or flat trees. The reference column assignments are given by aligning using the gold tags. We see that the full trees result in higher accuracy column assignments versus the use of flat trees, even when using an equivalent number of columns via the use of anchors. Only with 5 anchors do the accuracies converge. Interestingly, the overlap of the column assignments when 5 anchors are used remains below 50 percent – hence this convergence is not due to selection of the same anchors.

Anchor columns	Full tree derived PSSM	Flat tree derived PSSM
∞	90.2	88.8
100	90.2	88.9
10	93.0	92.0
5	95.0	95.0

Table 3: Accuracy of column assignments when given POS-tags, section 24

Tags	Baseline	Full tree derived PSSM	Flat tree derived PSSM
gold	90.1	90.4	90.0
tagger	89.2	89.5	89.1

Table 4: Base phrase bracketing F-measure accuracy, section 24

To incorporate column assignments into base phrase chunking, we included sequences of columns as features in our model, just as we have used features of POS-tags. Thus, for instance, in Table 2, the main verb (VB) is column 3 when 5 anchors are used to define coarse columns, and it is preceded by a column 2 MD. Hence features associated with a particular base phrase tag X would include

- (chunktag$_i$ = X; POS-tag$_{i-1}$ = MD; POS-tag$_i$ = VB)
- (chunktag$_i$ = X; Anchor5-tag$_{i-1}$ = 2; Anchor5-tag$_i$ = 3)
- (chunktag$_i$ = X; POS/Anchor5-tag$_{i-1}$ = MD/2; POS/Anchor5-tag$_i$ = VB/3)

For the current trials, the features included tritag base phrase tags, up to 4 POS-tag sequences, bitag column tags, and bitag (column $\times$ POS-tag) features. Our best results were achieved using coarse columns derived from 5 anchors. Table 4 shows our results for both reference (gold) tags and POS-tagger tags. While the improvements are modest for full tree derived PSSM column annotations, there is no gain when using the flat trees to derive the PSSM.

6 Discussion

The results presented here show modest improvements given PSSM assigned column sequences as features within a base phrase chunker, indicating at least some utility of these annotations beyond the tags themselves. The process by which the PSSM is constructed appears to be a key consideration, since the alignments based on flat trees did not yield annotations of utility for this task. The chosen method for deriving the PSSM represents essentially the first approach that we have taken, hence there remains much to investigate, and likely further improvements to the derived annotations.

We can also conclude that the definition of anchors to derive coarse regions of the string appears to be a useful. It could very well be the case that specifically building the MSA to preserve anchor columns would be an important consideration. For example, in the column assignments in Table 2, we see that the fifth token in the string 'NNS', which is within an appositive modifying the subject NP, occupies an anchor slot, as opposed to the subject head noun itself (which is the second token). This suggests that the iterative re-estimation of the PSSM is likely causing the column assignments to diverge from their original, tree-derived meanings. This also remains an important area of further research.

We would also like to continue working on the alignment model for deriving the column tags. We can likely do much better than the simple PSSM in assigning columns to input POS-tag sequences. Further, we could build joint POS-tagging and alignment models, perform forward-backward estimation and marginalize over the POS-tags to derive more robust estimates of the columns given a particular string. In addition, we would like to explore other methods for making use of column sequences as features within base phrase parsing models.

While there remains much work to be done to understand how best to approach this problem of MSA from syntactic trees, this report provides an initial indication that the information derived from such alignments can be of utility within finite-state syntactic models.

References

[1] Laura Alonso, Irene Castellon, Jordi Escribano, Xavier Messeguer, and Lluis Padro. Multiple sequence alignment for characterizing the linear structure of revision. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, 2004.

[2] G.J. Barton and M.J.E. Sternberg. A strategy for the rapid multiple alignment of protein sequences. *Journal of Molecular Biology*, 198:327–337, 1987.

[3] Regina Barzilay and Lillian Lee. Bootstrapping lexical choice via multiple-sequence alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 164–171, 2002.

[4] Regina Barzilay and Lillian Lee. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of the Human Language Technology Conference and Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 16–23, 2003.

[5] Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*, pages 132–139, 2000.

[6] Michael J. Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, 1997.

[7] Michael J. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–8, 2002.

[8] Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis*. Cambridge University Press, Cambridge, UK, 1998.

[9] Seeger Fisher and Brian Roark. The utility of parse-derived features for automatic discourse segmentation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 488–495, 2007.

[10] Kristy Hollingshead, Seeger Fisher, and Brian Roark. Comparing and combining finite-state and context-free parsers. In *Proceedings of the Conference on Human Language Technology Conference and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 787–794, 2005.

[11] David M. Magerman. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, 1995.

[12] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[13] Mehryar Mohri and Mark-Jan Nederhof. Regular approximation of context-free grammars through transformation. In Jean-Claude Junqua and Gertjan van Noord, editors, *Robustness in Language and Speech Technology*, pages 153–163. Kluwer Academic Publishers, Dordrecht, 2001.

[14] Mark-Jan Nederhof. Practical experiments with regular approximation of context-free languages. *Computational Linguistics*, 26(1):17–44, 2000.

[15] Fernando C. N. Pereira and Rebecca N. Wright. Finite-state approximation of phrase-structure grammars. In E. Roche and Y. Schabes, editors, *Finite-State Language Processing*, pages 149–173. MIT Press, Cambridge, MA, 1997.

[16] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of the Human Language Technology Conference and Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 213–220, 2003.

[17] Bangalore Srinivas and Aravind K. Joshi. Supertagging: an approach to almost parsing. *Computational Linguistics*, 25(2):237–265, 1999.