Storylines

October 28, 2020

1 Introduction

2 Background: Pairwise Alignment

Insertion and deletion versus compression and expansion

Sentence Representations

3 Problem Setup: Multiple Sequence Alignment

The goal of multiple sequence alignment (MSA) is to find a joint alignment of K sequences $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{K}$, where each element is a sequence of tokens: $\mathbf{x}_i = \langle x_{i1}, x_{i2}, \ldots \rangle, x_{ij} \in \Sigma$. Tokens from the alphabet can be a vector or the gap symbol: $\Sigma = \mathbb{R}^n \cup \{-\}$.

A multiple alignment is a matrix $M \in \Sigma^{K \times L}$, given by:

$$M = \begin{pmatrix} \tilde{\mathbf{x}}_1 \\ \vdots \\ \tilde{\mathbf{x}}_K \end{pmatrix}, \tag{1}$$

where each $\tilde{\mathbf{x}}_i$ is obtained by inserting gaps into the corresponding $\mathbf{x}_i$ to ensure that each $\tilde{\mathbf{x}}_i$ is of length L.

There are two common measures of quality for an alignment: 1) the sum of pairs (SP) score and 2) the consensus error or Steiner distance. In order to compute the SP score, we require a scoring function $d : \Sigma \times \Sigma \to \mathbb{R}$ that operates on pairs of tokens. The SP score is given by

$$\mathrm{SP}(M) = \sum_{t=1}^{L} \sum_{i=1}^{K} \sum_{j=i+1}^{K} d(M_{it}, M_{jt}), \tag{2}$$

obtained by summing the pairwise distances between elements of M in the same column.

The consensus error given a consensus sequence $\mathbf{z} \in \Sigma^L$ is defined as follows:

$$\mathrm{CE}(M, \mathbf{z}) = \sum_{t=1}^{L} \sum_{i=1}^{K} d(M_{it}, z_t), \tag{3}$$

the sum of the distances from each element in a column to z_t. The mean sequence $\mathbf{z} \in \Sigma^L$ that minimizes this error is known as the consensus sequence.

Finally, the Steiner distance is closely related to the consensus error, and can be computed without explicitly constructing a multiple alignment. Let $D : \Sigma^* \times \Sigma^* \to \mathbb{R}$ be the distance measure obtained under global pairwise alignment using the token distance d, so that D operates on pairs of sequences. The Steiner distance is given by

$$\mathrm{SD}(\mathcal{X}, \mathbf{z}) = \sum_{i=1}^{K} D(\mathbf{x}_i, \mathbf{z}). \tag{4}$$

The optimal consensus error, $\min_{\mathbf{z}} \mathrm{CE}(M, \mathbf{z})$, is equivalent to the optimal Steiner distance, $\min_{\mathbf{z}} \mathrm{SD}(\mathcal{X}, \mathbf{z})$ [4]. Additionally, the optimal Steiner sequence, $\arg\min_{\mathbf{z}} \mathrm{SD}(\mathcal{X}, \mathbf{z})$, is equivalent to the optimal consensus sequence, $\arg\min_{\mathbf{z}} \mathrm{CE}(M, \mathbf{z})$, up to spaces [4].

We focus on optimizing the Steiner distance due to its computational benefits, but also perform evaluation using the Sum of Pairs score.

4 Methods

As optimizing all of the objectives in the previous section is NP-hard, we resort to heuristic optimization methods. We use a combination of three approaches: a progressive alignment method inspired by a classical algorithm from computation biology, an iterative averaging method similar to a popular algorithm from time-series analysis, and a greedy hill climbing algorithm that operates in the space of Steiner sequences.

4.1 Progressive Alignment

Inspired by the progressive alignment algorithm of Feng and Doolittle [3], we first take a naive approach to approximating a multiple sequence alignment. Progressive alignments serve to optimize the SP score.

Given a set of sequences $\mathcal{X}$ and an ordering σ, we progressively align the next sequence in the ordering to the already aligned sequences. Once a set of sequences are aligned, their columns of the alignment are frozen; elements of new sequences must align to a whole column from the existing alignment or a gap. This is referred to as the 'once a gap, always a gap' property [].

In order to align a sequence to an existing alignment, we lift the definition of the token distance d to compare an element to a column of an alignment such that $d^+ : \Sigma^* \times \Sigma \to \mathbb{R}$, as follows:

$$d^+(x, \mathbf{y}) = \sum_{j=1}^{|\mathbf{y}|} d(y_j, x). \tag{5}$$

We can then extend pairwise global alignment with d^+, allowing us to treat the columns of an alignment as if it were a token. With the extended measure d^+, the full algorithm is given in Algorithm 1.

Algorithm 1 Progressive Alignment

```
Given: A set of sequences $\mathcal{X} = \{\mathbf{x}_i\}_i$, ordering $\sigma$, and extended distance measure $d^+$
Initialize alignment $M$ to $\mathbf{x}_{\sigma(1)}$
for all sequences $\mathbf{x}_{\sigma(i)}$ in order $\sigma$ do
    Update $M = \textsc{PairwiseAlign}(M, \mathbf{x}_{\sigma(i)}, d^+)$
return $M$
```

4.2 An Iterative Averaging Algorithm

As the progressive algorithm does not minimize the Steiner distance, we propose an iterative averaging (IA) algorithm which directly optimizes the Steiner distance. The IA algorithm is inspired by the DTW Barycenter Averaging algorithm [5], which iteratively computes pairwise alignments between a mean sequence $\mathbf{z}$ and a set of sequences $\mathcal{X}$ then uses those alignments to recompute the mean sequence.

The IA algorithm proceeds in the largely same fashion; however, there is nuance in the second step of recomputing the mean sequence. We recompute the mean sequence by first constructing a multiple alignment from the pairwise alignments, then for each column of the multiple alignment computing the token that minimizes the distance to each element within that column. As we focus on insertion/deletion rather than compression/expansion (write about this in background), there is ambiguity when mapping the pairwise alignments to a joint multiple alignment. Namely, given a token from $\mathbf{x}_i$ aligned to a gap, there are multiple ways of that token to tokens from other sequences $\mathbf{x}_j$ that are also aligned to gaps. We resolve this ambiguity heuristically by inserting gaps in the $\tilde{\mathbf{x}}_i$ obtained by pairwise alignment with $\mathbf{z}$ so that all tokens aligned to elements in $\mathbf{z}$ are in the same column. (Need to edit for clarity, and background on pairwise alignment)

4.3 Hill Climbing Algorithm

As the previous iterative algorithm used a heuristic in the averaging step that was computationally inexpensive but not guaranteed to improve the Steiner distance, we also consider a greedy hill climbing (HC) algorithm that is more expensive but guaranteed to improve the objective.

The hill climbing algorithm proceeds as follows: Given an initial mean sequence $\mathbf{z}^{(0)}$, we first compute the pairwise alignments from $\mathbf{z}$ to each sequence in $\mathcal{X}$. We obtain the first proposal sequence by averaging the token from each $\mathbf{x}_i$ aligned to each z_t. Let this proposal be $\mathbf{z}'$. Next, we consider all proposals

Algorithm 2 Iterative Averaging Alignment

```
Given: A set of sequences $\mathcal{X} = \{\mathbf{x}_i\}_i$, and extended distance measure $d^+$
Initialize mean string $\mathbf{z}$
for all iterations do
    for all sequences $\mathbf{x}_i \in \mathcal{X}$ do
        Compute pairwise alignments $M_i = \textsc{PairwiseAlign}(\mathbf{x}_i, \mathbf{z}, d)$
    Stack the alignments $M = \textsc{Stack}(M_1, \ldots, M_K)$
    Update $\mathbf{z} = \textsc{Average}(M)$
return $M$
```

obtainable from one-step deviations from $\mathbf{z}'$, i.e. by adding or removing a single token. To obtain the removal proposals, we simply consider removing each element of $\mathbf{z}'$. For the addition proposals, for each we take the average of the cartesian-product of tokens from each $\mathbf{x}_i$ aligned to a gap

Algorithm 3 Hill Climbing Alignment

```
Given: A set of sequences $\mathcal{X} = \{\mathbf{x}_i\}_i$, and extended distance measure $d^+$
Initialize mean string $\mathbf{z}$
for all iterations do
    for all sequences $\mathbf{x}_i \in \mathcal{X}$ do
        Compute pairwise alignments $M_i = \textsc{PairwiseAlign}(\mathbf{x}_i, \mathbf{z}, d)$
    compute proposal $\mathbf{z}' = \textsc{AverageAligned}(M_1, \ldots, M_{|\mathcal{X}|})$

    for all  do
    for all  do
    Update $\mathbf{z} = \textsc{Average}(M)$
return $M$
```

5 Experiments

We evaluate our MSA approaches on the WritingPrompts dataset [2], a dataset of 300K human-written short stories obatined from the WritingPrompts subreddit. Each story consists of a pair of a writing prompt and the story itself. In our experiments, we use only the story.

We perform MSA on story sets of size 5. As it would be intractable to run MSA on all story sets of size 5, we use heuristics to select clusters of stories that have low pairwise distance. For every story in the WritingPrompts dataset, we encode each sentence using SBERT [6]. We then compute bigram bag-of-sentence (BoS) representations of stories by concatenating the SBERT representations of consecutive sentences, and averaging over time. Then, taking each of the first 50k stories from the WritingPrompts dataset as centroids, we find the 128 nearest neighbours for each of those centroids in bigram BoS space. We then find the 4 closest stories to each centroid from its 128 neighbours under the path-length normalized pairwise alignment distance. When finding the 4 closest stories, we additionally ensured that each of the 4 stories had a normalized pairwise distance of at least 150 in order to avoid duplicate stories. Finally, we then run MSA on the 50 centroids and their 4 closest neighbours with the lowest sum of normalized pairwise distances from the centroid to the neighbours.

For the pairwise alignment distance measure $d(x, y)$, with tokens $x, y \in \Sigma$, we use

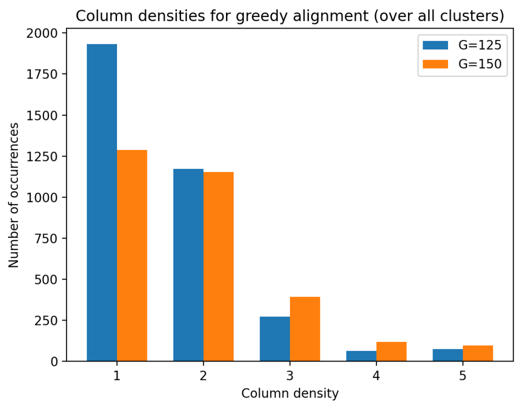
$$d(x,y) = \begin{cases} 0 & x = - \wedge y = - \\ \delta_x & x = - \wedge y \neq - \\ \delta_y & x \neq - \wedge y = - \\ \|x - y\|_2^2 & \text{otherwise,} \end{cases} \tag{6}$$

where δ_x, δ_y are the gap penalties. In choosing clusters, as well as our experiments with progressive alignment, we set $\delta_x = \delta_y \in \{125, 150\}$. For the Steiner distance, we set $\delta_x \in \{60, 75\}$ and $\delta_y \in \{180, 225, 300, 375\}$. We chose these gap penalties empirically based on preliminary analysis of the SBERT nearest neighbours of sentences as well as the resulting multiple alignments.

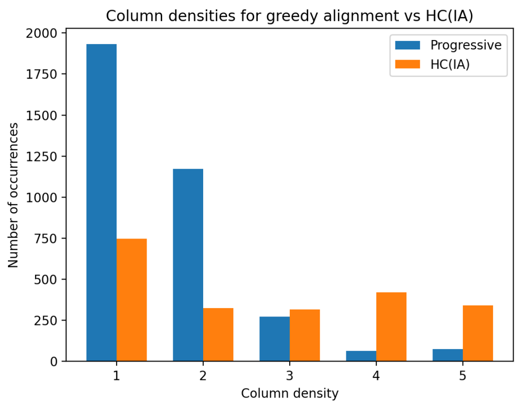
We compare multiple alignments based on the SP score and Steiner distance, and examine the output of each MSA algorithm by qualitatively evaluating the semantic closeness of alignments. For computing

Algorithm	δ_x	δ_y	SP Score	Steiner Distance
Progressive	125	125	2,494,153.00	877,445.99
Hill Climbing (Progressive)	60	300	2,514,774.25	774,525.85
Iterative Averaging	60	300	2,643,249.25	716,350.33
Hill Climbing (IA)	60	300	2,647,854.25	709,076.44

Table 1: The SP scores and Steiner distances for each of the MSA algorithms. The progressive algorithm optimizes the SP score and achieves the lowest. The hill climbing algorithm initialized with the mean sequence obtained from IA obtains the lowest Steiner distance. We see the algorithms get stuck in local optima, as the hill climbing algorithm initialize with the progressive mean sequence obtains a much higher Steiner distance than if it had been initialized with IA.



(a)



(b)

Figure 1: Merge into one graph.

the MSA, we use the progressive alignment, iterative averaging, and hill climbing algorithm. For the IA algorithm, we initialize the mean sequence with the longest sequence $\mathbf{x}_i \in \mathcal{X}$. For the hill climbing algorithm, we initialize the mean sequence with either the mean sequence obtained from the progressive alignment or the IA algorithm, yielding two configurations.

6 Results

We find that the MSA algorithms perform better on the objective they optimize, as seen in Table 1. The progressive alignment outperforms the other approaches by getting a lower SP score, while the IA and hill climbing (initialized with the IA mean sequence) approaches perform well on the Steiner distance. Initializing the hill climbing approach with the mean sequence obtained from the progressive alignment results in an alignment that has a lower Steiner distance than progressive, but the SP score increases.

The alignments obtained from the Progressive alignments are longer and contain more gaps than the hill climbing (IA) alignments. This is shown in Figure 1, where the total number of columns is much larger for progressive than HC(IA), and the column densities are lower for progressive as well. We hypothesize that this is due to the algorithms chosen for optimizing the Steiner distance. In the IA algorithm, the gap penalty controls how the mean sequence accounts for the number of gaps within a column. JK, need to think more about this still [1].

References

[1] Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, page 792–803. VLDB Endowment, 2004. ISBN 0120884690.

[2] Angela Fan, Mike Lewis, and Yann N. Dauphin. Hierarchical neural story generation. *CoRR*, abs/1805.04833, 2018. URL `http://arxiv.org/abs/1805.04833`.

[3] DF Feng and RF Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. 1987.

[4] Dan Gusfield. *Multiple String Comparison – The Holy Grail*, page 332–369. Cambridge University Press, 1997. doi: 10.1017/CBO9780511574931.017.

[5] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.

[6] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL `https://www.aclweb.org/anthology/D19-1410`.