Storylines

October 29, 2020

1 Introduction

2 Background: Pairwise Sequence Alignment

The goal of pairwise sequence alignment is to find a minimum cost alignment between sequences of sentences $\mathbf{x}_1$ and $\mathbf{x}_2 \in \Sigma^*$, where sentences are represented by a vector in $\mathbb{R}^n$. We additionally add a gap symbol '-' to Σ, such that $\Sigma = \mathbb{R}^n \cup \{-\}$. We refer to elements of Σ as tokens.

In order to align two stories, pairwise alignment extends edit distance with a distance measure $d : \Sigma \times \Sigma \to \mathbb{R}$ that operates on pairs of tokens, one from each story. The best alignment is given by the following:

$$\underset{\pi}{\operatorname{argmin}} \sum_{(i,j)\in\pi} d(x_{1,i}, x_{2,j}) \tag{1}$$

where π is a joint path through $\mathbf{x}_1$ and $\mathbf{x}_2$. Valid paths are obtained by only inserting gaps into the original story while preserving order. Equation 1 can be solved exactly via dynamic programming. Let $\mathbf{x}_1^\pi, \mathbf{x}_2^\pi$ be the gap-augmented stories obtained by following path π, which are both of length L. We refer to $M = \begin{pmatrix} \mathbf{x}_1^\pi \\ \mathbf{x}_2^\pi \end{pmatrix}$ as the alignment.

When choosing a distance measure d, a key component is how the measure deals with gaps. There are two approaches: 1) gaps correspond to insertion or deletions and 2) gaps correspond to expansions or compressions. Classical algorithms from biology, such as Needleman-Wunsch, model insertions and deletions, as random mutations result in completely new elements that do not have a good interpretation as an expansion or compression. Time-series algorithms, such as Dynamic Time Warping (DTW), model expansions and compressions, since warping may occur due to sampling at different frequencies or other reasons. It is possible to model both of these in the distance measure. For simplicity, we focus only on modeling insertion and deletion as this is a good fit for story alignment: Adding a sentence to a story, even if it builds on prior sentences, adds new information.

3 Problem Setup: Multiple Sequence Alignment

In order to extract storylines, we would like to find patterns that are robust across multiple stories. Unfortunately, methods for solving pairwise alignment are not directly applicable, since they only consider pairs of stories. Instead, we turn to the problem of multiple sequence alignment, which generalizes alignment from pairs of stories to sets of stories.

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

$$M = \begin{pmatrix} \mathbf{x}_1^\pi \\ \vdots \\ \mathbf{x}_K^\pi \end{pmatrix}, \tag{2}$$

where each $\mathbf{x}_i^\pi$ is obtained by inserting gaps into the corresponding $\mathbf{x}_i$ to ensure that each $\mathbf{x}_i^\pi$ is of length L, as in pairwise alignment.

There are two common measures of quality for a multiple 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

$$\mathrm{SP}(M) = \sum_{l=1}^{L} \sum_{i=1}^{K} \sum_{j=i+1}^{K} d(M_{il}, M_{jl}), \tag{3}$$

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:

$$\mathrm{CE}(M, \mathbf{z}) = \sum_{l=1}^{L} \sum_{i=1}^{K} d(M_{il}, z_l), \tag{4}$$

the sum of the distances from each element in a column to z_l. 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

$$\mathrm{SD}(\mathcal{X}, \mathbf{z}) = \sum_{i=1}^{K} D(\mathbf{x}_i, \mathbf{z}). \tag{5}$$

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.

We utilize algorithms that optimize both the SP score and the Steiner distance, and report results on both.

4 Methods

As optimizing both the SP score and Steiner distance are 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 aim 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 [3].

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:

$$d^+(\mathbf{y}, x) = \sum_{j=1}^{|\mathbf{y}|} d(y_j, x). \tag{6}$$

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 $M$ = PairwiseAlign($M, \mathbf{x}_{\sigma(i)}, d^+$)
return $M$
```

4.2 An Iterative Averaging Algorithm

Next, we propose an iterative averaging (IA) algorithm which directly optimizes the Steiner distance. The IA algorithm is inspired by the DTW Barycenter Averaging (DBA) 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. As the DBA algorithm was designed to model expansion and compression, we adapt it for insertion and deletion.

The IA algorithm proceeds in a fashion similar to DBA. 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 model insertions and deletions, there is ambiguity when mapping the pairwise alignments to a joint multiple alignment. Namely, there are multiple ways of aligning tokens from $\mathcal{X}$ that are all to gaps. We resolve this ambiguity heuristically by inserting gaps in the $\mathbf{x}_i^\pi$ obtained by pairwise alignment with $\mathbf{z}$ so that all tokens aligned to elements in $\mathbf{z}$ are aligned. The algorithm is detailed in Algorithm 2.

Algorithm 2 Iterative Averaging Alignment

```
Given: A set of sequences $\mathcal{X} = \{\mathbf{x}_i\}_i$ and distance measure $d$
Given: Initial mean string $\mathbf{z}$
function IterativeAverage($\mathbf{z}, \mathcal{X}, d$)
    for all iterations do
        for all sequences $\mathbf{x}_i \in \mathcal{X}$ do
            Compute pairwise alignments $M^i = \text{PairwiseAlign}(\mathbf{x}_i, \mathbf{z}, d), M^i \in \Sigma^{2\times L_i}$
        Compute multiple alignment $M = \text{Stack}(M^1, \dots, M^K)$
        Update $\mathbf{z} = \text{Average}(M, d)$
    return $M$
function Stack($M^1, \dots, M^K$)
    while Non-gap elements of $\mathbf{z}$ are not aligned in all $M_2^i$ do
        Record column indices of $M_2^i$ that have a non-gap element of $\mathbf{z}$
        Insert $(-,-)$ columns to the left of the columns with non-gap elements from $\mathbf{z}$ for each $M^i$
    return $\begin{pmatrix} M_1^1 \\ \dots \\ M_1^K \end{pmatrix}$
function Average($M, d$)
    Initialize $\mathbf{z} \in \Sigma^L$
    for all columns $l$ in $M$ do
        Set proposal $z'$ to the average non-gap representation of column $l$
        Compute cost $c(z') = \sum_i d(M_{il}, z')$ and $c(-) = \sum_i d(M_{il}, -)$
        if $c(z) > c(-)$ then
            Set $z_l = -$
        else
            Set $z_l = z'$
    return $\mathbf{z}$
```

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 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 sets of stories that have low pairwise distance. We use the following procedure to obtain the sets $\mathcal{X}$:

1. For every story in the WritingPrompts dataset, we project each sentence using SBERT [6] into $\mathbb{R}^n$.
2. Compute bigram bag-of-sentence (BoS) representations of stories by concatenating the SBERT representations of consecutive sentences, and averaging over time.
3. Take each of the first 50k stories from the WritingPrompts dataset as centroids, and find 128 nearest neighbours for each in bigram BoS space.
4. Find the 4 closest stories to each centroid from its 128 neighbours under the path-length normalized pairwise alignment distance. We avoid selecting duplicates by discarding stories with matching 10-grams.
5. Select 50 centroids (and their closest stories), ensuring coverage over stories of different styles.

For the token distance measure $d(x, y)$, we use

$$d(x,y) = \begin{cases} 0 & x = - \wedge y = - \\ \delta_x & x = - \wedge y \neq - \\ \delta_y & x \neq - \wedge y = - \\ \|x - y\|_2^2 & \text{otherwise,} \end{cases} \tag{7}$$

where δ_x, δ_y are the gap penalties. This distance measure

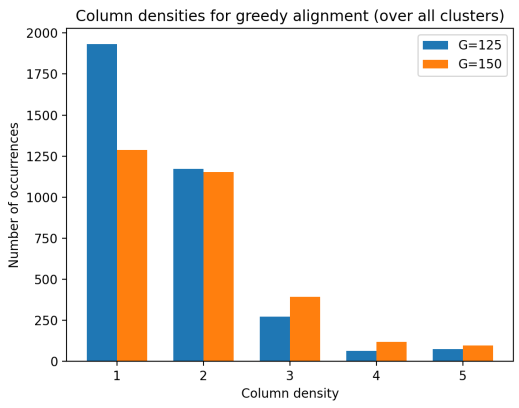
When choosing the distance measure d, there are two key points: 1) whether to model insertions and deletions or compressions and expansions, and 2) what token representation to operate on. For 1) we focus on insertions and deletions, as they are a better fit for aligning short stories about diverse topics. We are not concerned with identifying expansions upon the same story; we would like to find trends across somewhat different stories. For 2) we use

In choosing clusters, as well as our experiments with progressive alignment, we set $\delta_x = \delta_z \in \{125, 150\}$. 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.

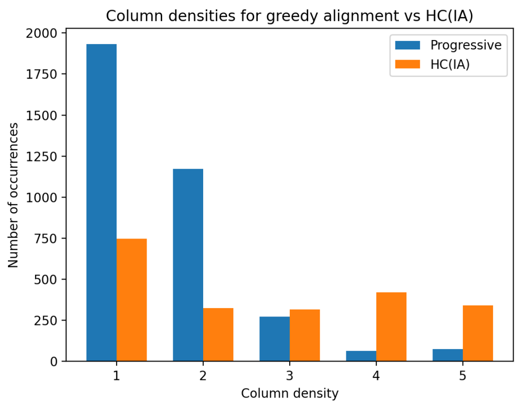
We compare multiple alignments based on the SP score and Steiner distance, and examine the output of each MSA algorithm by qualitatively evaluating the semantic closeness of alignments. For computing the MSA, we use the progressive alignment, iterative averaging, and hill climbing algorithm. For the 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.

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.

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`.