Text Alignment for Real-Time Crowd Captioning

Iftekhar Naim, **Daniel Gildea**, **Walter Lasecki** and **Jeffrey P. Bigham**
Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

The primary way of providing real-time captioning for deaf and hard of hearing people is to employ expensive professional stenographers who can type as fast as natural speaking rates. Recent work has shown that a feasible alternative is to combine the partial captions of ordinary typists, each of whom types part of what they hear. In this paper, we describe an improved method for combining partial captions into a final output based on weighted A* search and multiple sequence alignment (MSA). In contrast to prior work, our method allows the tradeoff between accuracy and speed to be tuned, and provides formal error bounds. Our method outperforms the current state-of-the-art on Word Error Rate (WER) (29.6%), BLEU Score (41.4%), and F-measure (36.9%). The end goal is for these captions to be used by people, and so we also compare how these metrics correlate with the judgments of 50 study participants, which may assist others looking to make further progress on this problem.

1 Introduction

Real-time captioning provides deaf or hard of hearing people access to speech in mainstream classrooms, at public events, and on live television. To maintain consistency between the captions being read and other visual cues, the latency between when a word was said and when it is displayed must be under five seconds. The most common approach to real-time captioning is to recruit a trained stenographer with a special purpose phonetic keyboard, who transcribes the speech to text within approximately 5 seconds. Unfortunately, professional captionists are quite expensive ($150 per hour), must be recruited in blocks of an hour or more, and are difficult to schedule on short notice. Automatic speech recognition (ASR) (Saraclar et al., 2002) attempts to solve this problem by converting speech to text completely automatically. However, the accuracy of ASR quickly plummets to below 30% when used on an untrained speaker's voice, in a new environment, or in the absence of a high quality microphone (Wald, 2006b).

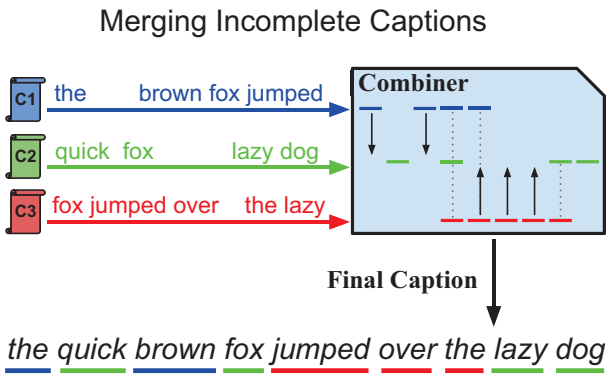


Figure 1: General layout of crowd captioning systems. Captionists (C1, C2, C3) submit partial captions that are automatically combined into a high-quality output.

An alternative approach is to combine the efforts of multiple non-expert captionists (anyone who can type) (Lasecki et al., 2012; Lasecki and Bigham, 2012; Lasecki et al., 2013). In this approach, multiple non-expert human workers transcribe an audio stream containing speech in real-time, and their partial input is combined to produce a final transcript (see Figure 1). This approach has been shown to dramatically outperform ASR in terms of both accuracy and Word Error Rate (WER), even when using captionists drawn from Amazon's Mechanical Turk. Furthermore, recall approached and even exceeded that of a trained expert stenographer with seven workers contributing, suggesting that the information is present to meet the performance of a stenographer. However, combining these captions involves real-time alignment of partial captions that may be incomplete and that often have spelling errors and inconsistent timestamps. In this paper, we present a more accurate combiner that leverages

Proceedings of NAACL-HLT 2013, pages 201–210,
Atlanta, Georgia, 9–14 June 2013.

Multiple Sequence Alignment (MSA) and Natural Language Processing to improve performance.

Gauging the quality of captions is not easy. Although word error rate (WER) is commonly used in speech recognition, it considers accuracy and completeness, not readability. As a result, a lower WER does not always result in better understanding (Wang et al., 2003). We compare WER with two other commonly used metrics: BLEU (Papineni et al., 2002) and F-measure (Melamed et al., 2003), and report their correlation with that of 50 human evaluators.

The key contributions of this paper are as follows:

- We have implemented an A*-search based Multiple Sequence Alignment algorithm (Lermen and Reinert, 2000) that can trade-off speed and accuracy by varying the heuristic weight and chunk-size parameters. We show that it outperforms previous approaches in terms of WER, BLEU score, and F-measure.
- We propose a beam-search based technique using the timing information of the captions that helps to restrict the search space and scales effectively to align longer sequences efficiently.
- We evaluate the correlation of WER, BLEU, and F-measure with 50 human ratings of caption readability, and found that WER was more highly correlated than BLEU score (Papineni et al., 2002), implying it may be a more useful metric overall when evaluating captions.

2 Related Work

Most of the previous research on real-time captioning has focused on Automated Speech Recognition (ASR) (Saraclar et al., 2002; Cooke et al., 2001; Pražák et al., 2012). However, experiments show that ASR systems are not robust enough to be applied for arbitrary speakers and in noisy environments (Wald, 2006b; Wald, 2006a; Bain et al., 2005; Bain et al., 2012; Cooke et al., 2001).

2.1 Crowd Captioning

To address these limitations of ASR-based techniques, the Scribe system collects partial captions from the crowd and then uses a graph-based incremental algorithm to combine them on the fly (Lasecki et al., 2012). The system incrementally builds a chain graph, where each node represents a set of equivalent words entered by the workers and the link between nodes are adjusted according to the order of the input words. A greedy search is performed to identify the path with the highest confidence, based on worker input and an n-gram language model. The algorithm is designed to be used online, and hence has high speed and low latency. However, due to the incremental nature of the algorithm and due to the lack of a principled objective function, it is not guaranteed to find the globally optimal alignment for the captions.

2.2 Multiple Sequence Alignment

The problem of aligning and combining multiple transcripts can be mapped to the well-studied Multiple Sequence Alignment (MSA) problem (Edgar and Batzoglou, 2006). MSA is an important problem in computational biology (Durbin et al., 1998). The goal is to find an optimal alignment from a given set of biological sequences. The pairwise alignment problem can be solved efficiently using dynamic programming in $O(N^2)$ time and space, where N is the sequence length. The complexity of the MSA problem grows exponentially as the number of sequences grows, and has been shown to be NP-complete (Wang and Jiang, 1994). Therefore, it is important to apply some heuristic to perform MSA in a reasonable amount of time.

Most MSA algorithms for biological sequences follow a progressive alignment strategy that first performs pairwise alignment among the sequences, and then builds a guide tree based on the pairwise similarity between these sequences (Edgar, 2004; Do et al., 2005; Thompson et al., 1994). Finally, the input sequences are aligned according to the order specified by the guide tree. While not commonly used for biological sequences, MSA with A*-style search has been applied to these problems by Horton (1997) and Lermen and Reinert (2000).

Lasecki et al. explored MSA in the context of merging partial captions by using the off-the-shelf MSA tool *MUSCLE* (Edgar, 2004), replacing the nucleotide characters by English characters (Lasecki et al., 2012). The substitution cost for nucleotides was replaced by the 'keyboard distance' between English characters, learned from the physical layout of a keyboard and based on common spelling

errors. However, MUSCLE relies on a progressive alignment strategy and may result in suboptimal solutions. Moreover, it uses characters as atomic symbols instead of words. Our approach operates on a per-word basis and is able to arrive at a solution that is within a selectable error-bound of optimal.

3 Multiple Sequence Alignment

We start with an overview of the MSA problem using standard notations as described by Lermen and Reinert (2000). Let $S_1, \ldots, S_K, K \geq 2$, be the K sequences over an alphabet Σ, and having length $N_1, \ldots, N_K$. The special gap symbol is denoted by '$-$' and does not belong to Σ. Let $A = (a_{ij})$ be a $K \times N_f$ matrix, where $a_{ij} \in \Sigma \cup \{-\}$, and the i^{th} row has exactly $(N_f - N_i)$ gaps and is identical to S_i if we ignore the gaps. Every column of A must have at least one non-gap symbol. Therefore, the j^{th} column of A indicates an alignment state for the j^{th} position, where the state can have one of the $2^K - 1$ possible combinations. Our goal is to find the optimum alignment matrix A_{OPT} that minimizes the sum of pairs (SOP) cost function:

$$c(A) = \sum_{1 \leq i \leq j \leq K} c(A_{ij}) \qquad (1)$$

where $c(A_{ij})$ is the cost of the pairwise alignment between S_i and S_j according to A. Formally, $c(A_{ij}) = \sum_{l=1}^{N_f} \mathrm{sub}(a_{il}, a_{jl})$, where $\mathrm{sub}(a_{il}, a_{jl})$ denotes the cost of substituting a_{jl} for a_{il}. If a_{il} and a_{jl} are identical, the substitution cost is usually zero. For the caption alignment task, we treat each individual word as a symbol in our alphabet Σ. The substitution cost for two words is estimated based on the edit distance between two words. The exact solution to the SOP optimization problem is NP-Complete, but many methods solve it approximately. In this paper, we adapt weighted A* search for approximately solving the MSA problem.

3.1 A* Search for MSA

The problem of minimizing the SOP cost function for K sequences is equivalent to estimating the shortest path between a single source and single sink node in a K-dimensional lattice. The total number of nodes in the lattice is $(N_1 + 1) \times (N_2 + 1) \times \cdots \times (N_K + 1)$, each corresponding to a distinct position in K sequences. The source node is $[0, \ldots, 0]$ and the sink node is $[N_1, \ldots, N_K]$. The dynamic programming algorithm for estimating the shortest path from source to sink treats each node position $[n_1, \ldots, n_K]$ as a state and calculates a matrix that has one entry for each node. Assuming the sequences have roughly same length N, the size of the dynamic programming matrix is $O(N^K)$. At each vertex, we need to minimize the cost over all its $2^K - 1$ predecessor nodes, and, for each such transition, we need to estimate the SOP objective function that requires $O(K^2)$ operations. Therefore, the dynamic programming algorithm has time complexity of $O(K^2 2^K N^K)$ and space complexity of $O(N^K)$, which is infeasible for most practical problem instances. However, we can efficiently solve it via heuristic A* search (Lermen and Reinert, 2000).

Algorithm 1 MSA-A* Algorithm

Require: K input sequences $\mathcal{S} = \{S_1, \ldots, S_K\}$ having length $N_1, \ldots, N_K$, heuristic weight w, beam size b

```
start ← 0^K, goal ← [N_1, ..., N_K]
g(start) ← 0, f(start) ← w × h(start).
Q ← {start}
while Q ≠ ∅ do
    n ← EXTRACT-MIN(Q)
    for all s ∈ {0,1}^K − {0^K} do
        n_i ← n + s
        if n_i = goal then
            Return the alignment matrix for the reconstructed
            path from start to n_i
        else if n_i ∉ Beam(b) then
            continue;
        else
            g(n_i) ← g(n) + c(n, n_i)
            f(n_i) ← g(n_i) + w × h(n_i)
            INSERT-ITEM(Q, n_i, f(n_i))
        end if
    end for
end while
```

We use A* search based MSA (shown in Algorithm 1, illustrated in Figure 2) that uses a priority queue Q to store dynamic programming states corresponding to node positions in the K dimensional lattice. Let $n = [n_1, \ldots, n_K]$ be any node in the lattice, s be the source, and t be the sink. The A* search can find the shortest path using a greedy Best First Search according to an evaluation function $f(n)$, which is the summation of the cost func-

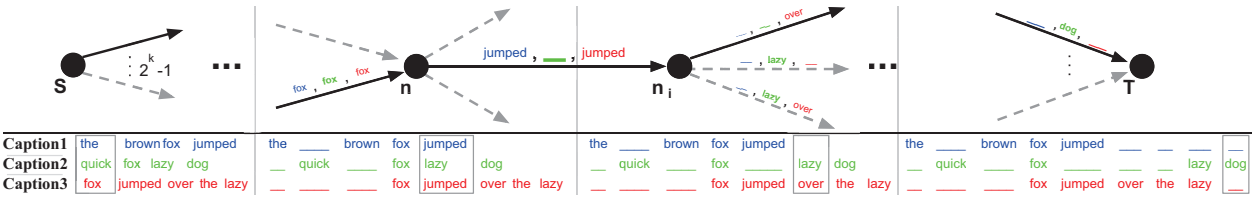


Figure 2: A^* MSA search algorithm. Each branch is one of $2^K - 1$ possible alignments for the current input. The branch with minimum sum of the current alignment cost and the expected heuristic value h_{pair} (precomputed).

tions $g(n)$ and the heuristic function $h(n)$ for node n. The cost function $g(n)$ denotes the cost of the shortest path from the source s to the current node n. The heuristic function $h(n)$ is the approximate estimated cost of the shortest path from n to the destination t. At each step of the A* search algorithm, we extract the node with the smallest $f(n)$ value from the priority queue Q and expand it by one edge. The heuristic function $h(n)$ is admissible if it never overestimates the cost of the cheapest solution from n to the destination. An admissible heuristic function guarantees that A* will explore the minimum number of nodes and will always find the optimal solution. One commonly used admissible heuristic function is $h_{pair}(n)$:

$$h_{pair}(n) = L(n \rightarrow t) = \sum_{1 \leq i < j \leq K} c(A_p^*(\sigma_i^n, \sigma_j^n)) \quad (2)$$

where $L(n \rightarrow t)$ denotes the lower bound on the cost of the shortest path from n to destination t, A_p^* is the optimal pairwise alignment, and σ_i^n is the suffix of node n in the i-th sequence. A* search using the pairwise heuristic function h_{pair} significantly reduces the search space and also guarantees finding the optimal solution. We must be able to estimate $h_{pair}(n)$ efficiently. It may appear that we need to estimate the optimal pairwise alignment for all the pairs of suffix sequences at every node. However, we can precompute the dynamic programming matrix over all the pair of sequences (S_i, S_j) once from the backward direction, and then reuse these values at each node. This simple trick significantly speeds up the computation of $h_{pair}(n)$.

Despite the significant reduction in the search space, the A* search may still need to explore a large number of nodes, and may become too slow for real-time captioning. However, we can further improve the speed by following the idea of *weighted* A^* search (Pohl, 1970). We modify the evaluation function $f(n) = g(n) + h_{pair}(n)$ to a weighted evaluation function $f'(n) = g(n) + wh_{pair}(n)$, where $w \geq 1$ is a weight parameter. By setting the value of w to be greater than 1, we increase the relative weight of the estimated cost to reach the destination. Therefore, the search prefers the nodes that are closer to the destination, and thus reaches the goal faster. Weighted A* search can significantly reduce the number of nodes to be examined, but it also loses the optimality guarantee of the admissible heuristic function. We can trade-off between accuracy and speed by tuning the weight parameter w.

3.2 Beam Search using Time-stamps

The computational cost of the A* search algorithm grows exponentially with increase in the number of sequences. However, in order to keep the crowdsourced captioning system cost-effective, only a small number of workers are generally recruited at a time (typically $K \leq 10$). We, therefore, are more concerned about the growth in computational cost as the sequence length increases.

In practice, we break down the sequences into smaller chunks by maintaining a window of a given time interval, and we apply MSA only to the smaller chunks of captions entered by the workers during that time window. As the window size increases, the accuracy of our MSA based combining system increases, but so does the computational cost and latency. Therefore, it is important to apply MSA with a relatively small window size for real-time captioning applications. Another interesting application can be the offline captioning, for example, captioning an entire lecture and uploading the captions later.

For the offline captioning problem, we can focus less on latency and more on accuracy by aligning longer sequences. To restrict the search space from exploding with sequence length (N), we apply a beam constraint on our search space using the time stamps of each captioned words. For example, if we

1.	so	now	what	i	want	to		do	is		introduce	some	of	the					
2.			what	i			wanna	do	is		introduce	some	of	the		aspects	of	the	class
3.	so	now	what	i	want	to		do	is	is	introduce	some	of	the		aspects	of	the	class
4.	so	now	what	i	want	to		do	is		introduce								
5.	so	now	what	i	want	to		do	is		introduce	some	of	the	operational		of	the	class
6.	so			i	want	to					introduce	some	of	the	operational	aspects	of	the	clas
C.	so	now	what	i	want	to		do	is		introduce	some	of	the	operational	aspects	of	the	class

Figure 3: An example of applying MSA-A* (threshold $t_v = 2$) to combine 6 partial captions (first 6 lines) by human workers to obtain the final output caption (C).

set the beam size to be 20 seconds, then we ignore any state in our search space that aligns two words having more than 20 seconds time lag. Given a fixed beam size b, we can restrict the number of priority queue removals by the A* algorithm to $O(Nb^K)$. The maximum size of the priority queue is $O(Nb^K)$. For each node in the priority queue, for each of the $O(2^K)$ successor states, the objective function and heuristic estimation requires $O(K^2)$ operations and each priority queue insertion requires $O(\log(Nb^K))$ i.e. $O(\log N + K \log b)$ operations. Therefore, the overall worst case computational complexity is $O\left(Nb^K 2^K (K^2 + \log N + K \log b)\right)$. Note that for fixed beam size b and number of sequences K, the computational cost grows as $O(N \log N)$ with the increase in N. However, in practice, weighted A* search explores much smaller number of states compared to this beam-restricted space.

3.3 Majority Voting after Alignment

After aligning the captions by multiple workers in a given chunk, we need to combine them to obtain the final caption. We do that via majority voting at each position of the alignment matrix containing a non-gap symbol. In case of tie, we apply the language model to choose the most likely word.

Often workers type in nonstandard symbols, abbreviations, or misspelled words that do not match with any other workers' input and end up as a single word aligned to gaps in all the other sequences. To filter out such spurious words, we apply a voting threshold (t_v) during majority voting and filter out words having less than t_v votes. Typically we set t_v = 2 (see the example in Figure 3). While applying the voting threshold improves the word error rate and readability, it runs the risk of loosing correct words if they are covered by only a single worker.

3.4 Incorporating an N-gram Language Model

We also experimented with a version of our system designed to incorporate the score from an n-gram language model into the search. For this purpose, we modified the alignment algorithm to produce a hypothesized output string as it moves through the input strings, as opposed to using voting to produce the final string as a post-processing step. The states for our dynamic programming are extended to include not only the current position in each input string, but also the last two words of the hypothesis string (i.e. $[n_1, \ldots, n_K, w_{i-1}, w_{i-2}]$) for use in computing the next trigram language model probability. We replace our sum-of-all-pairs objective function with the sum of the alignment cost of each input with the hypothesis string, to which we add the log of the language model probability and a feature for the total number of words in the hypothesis. Mathematically, we consider the hypothesis string to be the 0th row of the alignment matrix, making our objective function:

$$c(A) = \sum_{1 \le i \le K} c(A_{0,i}) + w_{len} \sum_{l=1}^{N_f} I(a_{0,l} \neq -) + w_{lm} \sum_{l=1}^{N_f} \log P(a_{0,l} | a_{0,l-2}, a_{0,l-1})$$

where w_{lm} and w_{len} are negative constants indicating the relative weights of the language model probability and the length penalty.

Extending states with two previous words results in a larger computational complexity. Given K sequences of length N each, we can have $O(NK)$ distinct words. Therefore, the number distinct states is $O(Nb^K(NK)^2)$ i.e. $O(N^3K^2b^K)$. Each state can have $O(K2^K)$ successors, giving an overall computational complexity of $O(N^3K^3b^K2^K(K^2 + \log N + \log K + K \log b))$. Alternatively, if the vo-

cabulary size $|V|$ is smaller than NK, the number of distinct states is bounded by $O(Nb^K|V|^2)$.

3.5 Evaluation Metric for Speech to Text Captioning

Automated evaluation of speech to text captioning is known to be a challenging task (Wang et al., 2003). Word Error Rate (WER) is the most commonly used metric that finds the best pairwise alignment between the candidate caption and the ground truth reference sentence. WER is estimated as $\frac{S+I+D}{N}$, where S, I, and D is the number of incorrect word substitutions, insertions, and deletions required to match the candidate sentence with reference, and N is the total number of words in the reference. WER has several nice properties such as: 1) it is easy to estimate, and 2) it tries to preserve word ordering. However, WER does not account for the overall 'readability' of text and thus does not always correlate well with human evaluation (Wang et al., 2003; He et al., 2011).

The widely-used BLEU metric has been shown to agree well with human judgment for evaluating translation quality (Papineni et al., 2002). However, unlike WER, BLEU imposes no explicit constraints on the word ordering. BLEU has been criticized as an 'under-constrained' measure (Callison-Burch et al., 2006) for allowing too much variation in word ordering. Moreover, BLEU does not directly estimate recall, and instead relies on the brevity penalty. Melamed et al. (2003) suggest that a better approach is to explicitly measure both precision and recall and combine them via F-measure.

Our application is similar to automatic speech recognition in that there is a single correct output, as opposed to machine translation where many outputs can be equally correct. On the other hand, unlike with ASR, out-of-order output is frequently produced by our alignment system when there is not enough overlap between the partial captions to derive the correct ordering for all words. It may be the case that even such out-of-order output can be of value to the user, and should receive some sort of partial credit that is not possible using WER. For this reason, we wished to systematically compare BLEU, F-measure, and WER as metrics for our task.

We performed a study to evaluate the agreement of the three metrics with human judgment. We randomly extracted one-minute long audio clips from four MIT OpenCourseWare lectures. Each clip was transcribed by 7 human workers, and then aligned and combined using four different systems: the graph-based system, and three different versions of our weighted A* algorithm with different values of tuning parameters. Fifty people participated in the study and were split in two equal sized groups. Each group was assigned two of the four audio clips, and each person evaluated all four captions for both clips. Each participant assigned a score between 1 to 10 to these captions, based on two criteria: 1) the overall estimated agreement of the captions with the ground truth text, and 2) the readability and understandability of the captions.

Metric	Spearman Corr.	Pearson Corr.
1-WER	0.5258	0.6282
BLEU	0.3137	0.6181
F-measure	0.4389	0.6240

Table 1: The correlation of average human judgment with three automated metrics: 1-WER, BLEU, and F-measure.

Finally, we estimated the correlation coefficients (both Spearman and Pearson) for the three metrics discussed above with respect to the average score assigned by the human participants. The results are presented in Table 1. Among the three metrics, WER had the highest agreement with the human participants. This indicates that reconstructing the correct word order is in fact important to the users, and that, in this aspect, our task has more of the flavor of speech recognition than of machine translation.

4 Experimental Results

We experiment with the MSA-A* algorithm for captioning different audio clips, and compare the results with two existing techniques. Our experimental set up is similar to the experiments by Lasecki et al. (2012). Our dataset consists of four 5-minute long audio clips extracted from lectures available on MIT OpenCourseWare. The audio clips contain speech from electrical engineering and chemistry lectures. Each audio clip is transcribed by ten non-expert human workers in real-time. We then combine these inputs using our MSA-A* algorithm, and also compare with the existing graph-based system and mul-

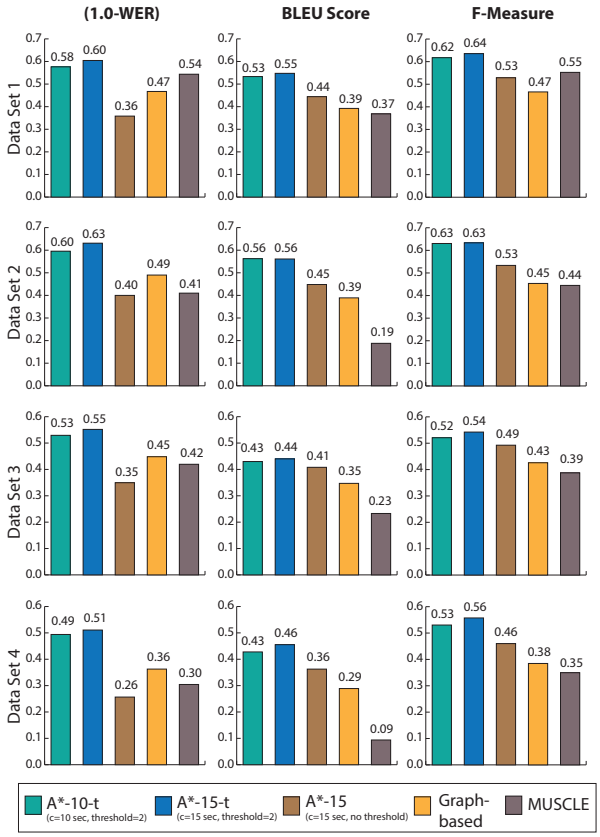


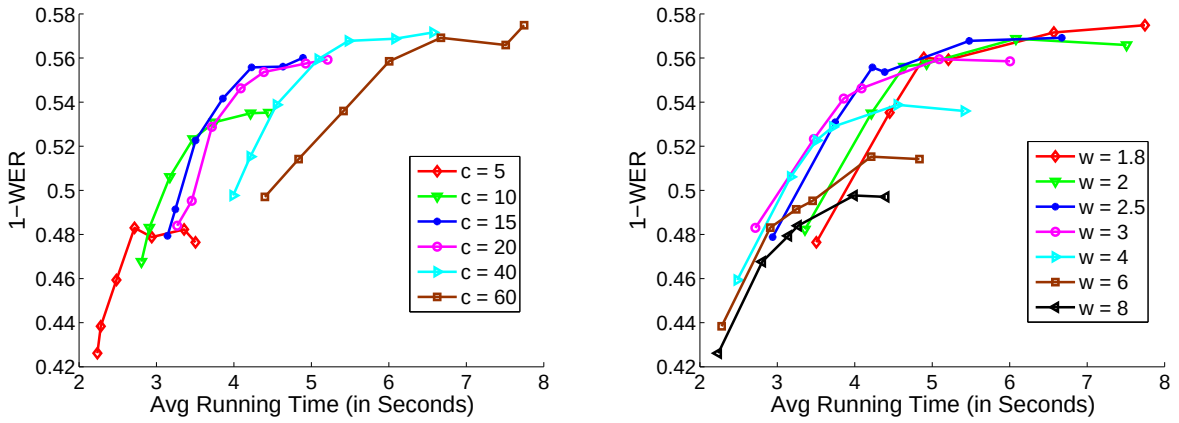
Figure 4: Evaluation of different systems on using three different automated metrics for measuring transcription quality: 1- Word Error Rate (WER), BLEU, and F-measure on the four audio clips.

tiple sequence alignment using MUSCLE.

As explained earlier, we vary the four key parameters of the algorithm: the chunk size (c), the heuristic weight (w), the voting threshold (t_v), and the beam size (b). The heuristic weight and chunk size parameters help us to trade-off between speed versus accuracy; the voting threshold t_v helps improve precision by pruning words having less than t_v votes, and beam size reduces the search space by restricting states to be inside a time window/beam. We use affine gap penalty (Edgar, 2004) with different gap opening and gap extension penalty. We set gap opening penalty to 0.125 and gap extension penalty to 0.05. We evaluate the performance using the three standard metrics: Word Error Rate (WER), BLEU, and F-measure. The performance in terms of these metrics using different systems is presented in Figure 4.

Out of the five systems in Figure 4, the first three are different versions of our A* search based MSA algorithm with different parameter settings: 1) A*-10-t system (c = 10 seconds, t_v = 2), 2) A*-15-t (c = 15 seconds, t_v = 2), and 3) A*-15 (c = 15 seconds, t_v = 1 i.e. no pruning while voting). For all three systems, the heuristic weight parameter w is set to 2.5 and beam size b = 20 seconds. The other two systems are the existing graph-based system and multiple sequence alignment using MUSCLE. Among the three A* based algorithms, both A*-15-t and A*-10-t produce better quality transcripts and outperform the existing algorithms. Both systems apply the voting threshold that improves precision. The system A*-15 applies no threshold and ends up producing many spurious words having poor agreement among the workers, and hence it scores worse in all the three metrics. The A*-15-t achieves 57.4% average accuracy in terms of (1-WER), providing 29.6% improvement with respect to the graph-based system (average accuracy 42.6%), and 35.4% improvement with respect to the MUSCLE-based MSA system (average accuracy 41.9%). On the same set of audio clips, Lasecki et al. (2012) reported 36.6% accuracy using ASR (Dragon Naturally Speaking, version 11.5 for Windows), which is worse than all the crowd-based based systems used in this experiment. To measure the statistical significance of this improvement, we performed a t-test at both the dataset level (n = 4 clips) and the word level (n = 2862 words). The improvement over the graph-based model was statistically significant with dataset level p-value 0.001 and word level p-value smaller than 0.0001. The average time to align each 15 second chunk with 10 input captions is $\sim$400 milliseconds.

We have also experimented with a trigram language model, trained on the British National Corpus (Burnard, 1995) having $\sim$122 million words. The language-model-integrated A* search provided a negligible 0.21% improvement in WER over the A*-15-t system on average. The task of combining captions does not require recognizing words; it only requires aligning them in the correct order. This could explain why language model did not improve accuracy, as it does for speech recognition. Since the standard MSA-A* algorithm (without language model) produced comparable accuracy and faster running time, we used that version in the rest of the



(a) Varying heuristic weights for fixed chunk sizes (c)

(b) Varying chunk size for fixed heuristic weight (w)

Figure 5: The trade-off between speed and accuracy for different heuristic weights and chunk size parameters.

experiments.

Next, we look at the critical speed versus accuracy trade-off for different values of the heuristic weight (w) and the chunk size (c) parameters. Since WER has been shown to correlate most with human judgment, we show the next results only with respect to WER. First, we fix the chunk size at different values, and then vary the heuristic weight parameter: w = 1.8, 2, 2.5, 3, 4, 6, and 8. The results are shown in Figure 5(a), where each curve represents how time and accuracy changed over the range of values of w and a fixed value of c. We observe that for smaller values of w, the algorithm is more accurate, but comparatively slower. As w increases, the search reaches the goal faster, but the quality of the solution degrades as well. Next, we fix w and vary chunk size c = 5, 10, 15, 20, 40, 60 second. We repeat this experiment for a range of values of w and the results are shown in Figure 5(b). We can see that the accuracy improves steeply up to $c = 20$ seconds, and does not improve much beyond $c = 40$ seconds. For all these benchmarks, we set the beam size (b) to 20 seconds and voting threshold (t_v) to 2.

In our tests, the beam size parameter (b) did not play a significant role in performance, and setting it to any reasonably large value (usually ≥ 15 seconds) resulted in similar accuracy and running time. This is because the A* search with h_{pair} heuristic already reduces the the search space significantly, and usually reaches the goal in a number of steps smaller than the state space size after the beam restriction.

Finally, we investigate how the accuracy of our algorithm varies with the number of inputs/workers. We start with a pool of 10 input captions for one of the audio clips. We vary the number of input captions (K) to the MSA-A* algorithm from 2 up to 10. The quality of input captions differs greatly among the workers. Therefore, for each value of K, we repeat the experiment $\min\left(20, \binom{10}{K}\right)$ times; each time we randomly select K input captions out of the total pool of 10. Figure 6 shows that accuracy steeply increases as the number of inputs increases to 7, and after that adding more workers does not provide much improvement in accuracy, but increases running time.

5 Discussion and Future Work

In this paper, we show that the A* search based MSA algorithm performs better than existing algorithms for combining multiple captions. The existing graph-based model has low latency, but it usually can not find a near optimal alignment because of its incremental alignment. Weighted A* search on the other hand performs joint multiple sequence alignment, and is guaranteed to produce a solution having cost no more than $(1 + \epsilon)$ times the cost of the optimal solution, given a heuristic weight of $(1 + \epsilon)$. Moreover, A* search allows for straightforward integration of an n-gram language model during the search.

Another key advantage of the proposed algorithm is the ease with which we can trade-off between

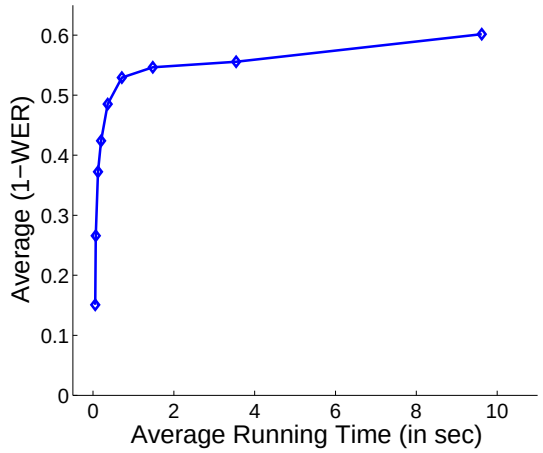


Figure 6: Experiments showing how the accuracy of the final caption by MSA-A* algorithm varies with the number of inputs from 2 to 10.

speed and accuracy. The algorithm can be tailored to real-time by using a larger heuristic weight. On the other hand, we can produce better transcripts for offline tasks by choosing a smaller weight.

It is interesting to compare our results with those achieved using the MUSCLE MSA tool of Edgar (2004). One difference is that our system takes a hierarchical approach in that it aligns at the word level, but also uses string edit distance at the letter level as a substitution cost for words. Thus, it is able to take advantage of the fact that individual transcriptions do not generally contain arbitrary fragments of words. More fundamentally, it is interesting to note that MUSCLE and most other commonly used MSA tools for biological sequences make use of a *guide tree* formed by a hierarchical clustering of the input sequences. The guide tree produced by the algorithms may or may not match the evolutionary tree of the organisms whose genomes are being aligned, but, nevertheless, in the biological application, such an underlying evolutionary tree generally exists. In aligning transcriptions, there is no particular reason to expect individual pairs of transcriptions to be especially similar to one another, which may make the guide tree approach less appropriate.

In order to get competitive results, the A* search based algorithm aligns sequences that are at least 7-10 seconds long. The delay for collecting the captions within a chunk can introduce latency, however, each alignment usually takes less than 300 milliseconds, allowing us to repeatedly align the stream of words, even before the window is filled. This provides less accurate but immediate response to users. Finally, when we have all the words entered in a chunk, we perform the final alignment and show the caption to users for the entire chunk.

After aligning the input sequences, we obtain the final transcript by majority voting at each alignment position, which treats each worker equally and does not take individual quality into account. Recently, some work has been done for automatically estimating individual worker's quality for crowd-based data labeling tasks (Karger et al., 2011; Liu et al., 2012). Extending these methods for crowd-based text captioning could be an interesting future direction.

6 Conclusion

In this paper, we have introduced a new A* search based MSA algorithm for aligning partial captions into a final output stream in real-time. This method has advantages over prior approaches both in formal guarantees of optimality and the ability to trade off speed and accuracy. Our experiments on real captioning data show that it outperforms prior approaches based on a dependency graph model and a standard MSA implementation (MUSCLE). An experiment with 50 participants explored whether existing automatic metrics of quality matched human evaluations of readability, showing WER did best.

Acknowledgments Funded by NSF awards IIS-1218209 and IIS-0910611.

References

Keith Bain, Sara Basson, A Faisman, and D Kanevsky. 2005. Accessibility, transcription, and access everywhere. *IBM Systems Journal*, 44(3):589–603.

Keith Bain, Eunice Lund-Lucas, and Janice Stevens. 2012. 22. transcribe your class: Using speech recognition to improve access for at-risk students. *Collected Essays on Learning and Teaching*, 5.

Lou Burnard. 1995. Users Reference Guide British National Corpus Version 1.0.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of bleu in machine translation research. In *Proceedings of EACL*, volume 2006, pages 249–256.

Martin Cooke, Phil Green, Ljubomir Josifovski, and Ascension Vizinho. 2001. Robust automatic speech recognition with missing and unreliable acoustic data. *Speech Communication*, 34(3):267–285.

Chuong B Do, Mahathi SP Mahabhashyam, Michael Brudno, and Serafim Batzoglou. 2005. Probcons: Probabilistic consistency-based multiple sequence alignment. *Genome Research*, 15(2):330–340.

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

Robert C Edgar and Serafim Batzoglou. 2006. Multiple sequence alignment. *Current opinion in structural biology*, 16(3):368–373.

Robert C Edgar. 2004. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797.

Xiaodong He, Li Deng, and Alex Acero. 2011. Why word error rate is not a good metric for speech recognizer training for the speech translation task? In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011*, pages 5632–5635. IEEE.

Phillip B Horton. 1997. *Strings, algorithms, and machine learning applications for computational biology*. Ph.D. thesis, University of California, Berkeley.

David R Karger, Sewoong Oh, and Devavrat Shah. 2011. Iterative learning for reliable crowdsourcing systems. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, volume 24, pages 1953–1961.

Walter Lasecki and Jeffrey Bigham. 2012. Online quality control for real-time crowd captioning. In *Proceedings of the 14th international ACM SIGACCESS conference on Computers and accessibility (ASSETS 2012)*, pages 143–150. ACM.

Walter Lasecki, Christopher Miller, Adam Sadilek, Andrew Abumoussa, Donato Borrello, Raja Kushalnagar, and Jeffrey Bigham. 2012. Real-time captioning by groups of non-experts. In *Proceedings of the 25rd annual ACM symposium on User interface software and technology, UIST '12*.

Walter Lasecki, Christopher Miller, and Jeffrey Bigham. 2013. Warping time for more effective real-time crowdsourcing. In *Proceedings of the ACM conference on Human Factors in Computing Systems*, CHI '13, page To Appear, New York, NY, USA. ACM.

Martin Lermen and Knut Reinert. 2000. The practical use of the A* algorithm for exact multiple sequence alignment. *Journal of Computational Biology*, 7(5):655–671.

Qiang Liu, Jian Peng, and Alex Ihler. 2012. Variational inference for crowdsourcing. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, volume 25, pages 701–709.

Dan Melamed, Ryan Green, and Joseph P Turian. 2003. Precision and recall of machine translation. In *Proceedings HLT-NAACL 2003*, volume 2, pages 61–63. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Ira Pohl. 1970. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3):193–204.

Aleš Pražák, Zdeněk Loose, Jan Trmal, Josef V Psutka, and Josef Psutka. 2012. Captioning of Live TV Programs through Speech Recognition and Respeaking. In *Text, Speech and Dialogue*, pages 513–519. Springer.

Murat Saraclar, Michael Riley, Enrico Bocchieri, and Vincent Goffin. 2002. Towards automatic closed captioning: Low latency real time broadcast news transcription. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, pages 1741–1744.

Julie D Thompson, Desmond G Higgins, and Toby J Gibson. 1994. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680.

Mike Wald. 2006a. Captioning for deaf and hard of hearing people by editing automatic speech recognition in real time. *Computers Helping People with Special Needs*, pages 683–690.

Mike Wald. 2006b. Creating accessible educational multimedia through editing automatic speech recognition captioning in real time. *Interactive Technology and Smart Education*, 3(2):131–141.

Lusheng Wang and Tao Jiang. 1994. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348.

Ye-Yi Wang, Alex Acero, and Ciprian Chelba. 2003. Is word error rate a good indicator for spoken language understanding accuracy. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003*, pages 577–582. IEEE.