Segment-Aware Dynamic Programming for Automatic Subtitle Reconstruction from ASR Transcription Errors

Ariel Cornelius Sitorus - 13524085
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: arielsitorus504@gmail.com , 13524085@std.stei.itb.ac.id

***Abstract*—Automatic subtitle reconstruction from Automatic Speech Recognition (ASR) output is complicated by the absence of sentence boundary markers in ASR transcripts, while reference scripts contain complete punctuation. Applying conventional sequence alignment directly to these two sources often produces cross-boundary word matches, resulting in incorrect word attribution and reduced reconstruction quality. This paper presents Segment-Aware Dynamic Programming (SADP), an extension of the Needleman-Wunsch algorithm that applies a cross-boundary penalty whenever a sentence boundary is detected by only one of the available boundary signals. Evaluation on 200 sentences from the Reuters News Corpus shows that SADP improves Word Attribution Accuracy (WAA) by 1.46 to 1.63 percent and reduces Segmentation Word Error Rate (SWER) by 1.47 to 2.31 percent compared with the standard Needleman-Wunsch algorithm. These improvements are achieved without increasing the algorithm's asymptotic time or space complexity.**

Keywords—dynamic programming; Needleman-Wunsch; subtitle reconstruction; ASR error correction; sentence boundary detection

I. Introduction

Automatic subtitle reconstruction aligns an ASR transcript to a pre-existing reference script so that each reference sentence is paired with its corresponding ASR words. The standard tool is the Needleman-Wunsch (NW) algorithm, a global sequence aligner based on dynamic programming. NW finds the globally optimal word-level alignment in $O(m \times n)$ time, but was designed for single-sequence alignment and has no concept of sentence boundaries. When applied to a multi-sentence passage, it may match an ASR word to the wrong sentence, particularly when the same content word recurs across consecutive sentences, a pattern common in topically coherent texts such as news articles.

Two boundary signals are available: script punctuation (sentence-end markers in the reference) and acoustic pauses (inter-word gaps in the ASR timestamp stream). This paper introduces Segment-Aware Dynamic Programming (SADP), a Needleman-Wunsch modification that adds a cross-boundary penalty Φ_x to DP transitions crossing a sentence boundary on exactly one of the two available signals (XOR condition), while maintaining $O(m \times n)$time and space complexity.

This paper makes four specific contributions,

- First, a formal problem statement for multi-sentence ASR-to-script global alignment with dual boundary signals is provided, separating the alignment task from downstream evaluation metrics.
- Second, the Segment-Aware Dynamic Programming (SADP) algorithm is introduced: a Needleman-Wunsch modification that adds a cross-boundary penalty to DP transitions crossing a sentence boundary on exactly one side (XOR logic), without increasing asymptotic complexity.
- Third, a provenance-aware evaluation framework is described, in which each ASR word carries a source-sentence tag assigned before noise injection, enabling direct measurement of word attribution accuracy.
- Fourth, an empirical evaluation on 400 sentences from the Reuters-21578 News Corpus is presented, covering three noise levels and a three-scenario ablation study that isolates the contribution of each boundary signal type.

II. Theoritical Background

A. Needleman-Wunsch Algorithm

The Needleman-Wunsch (NW) algorithm [1] is a dynamic programming algorithm for global sequence alignment. It was originally developed for biological sequence analysis and has since been applied to many alignment problems involving text, speech, and natural language processing. The algorithm aligns two complete sequences by finding the highest-scoring alignment over all possible combinations of matches, substitutions, insertions, and deletions.

Let the reference script be represented by the word sequence $R = (r_1, r_2, ..., r_m)$, and let the Automatic Speech Recognition (ASR) output be represented by the word sequence $A = (a_1, a_2,$

..., a_n). The alignment process begins by constructing a dynamic programming matrix with (m + 1) rows and (n + 1) columns. Each cell stores the best alignment score obtained for the corresponding prefixes of the two sequences.

For every matrix cell, three possible transitions are evaluated. The first transition aligns the current words from both sequences, producing either a match or a mismatch score. The second transition aligns a reference word with a gap, representing a deletion in the ASR output. The third transition aligns an ASR word with a gap, representing an insertion. The recurrence is defined as

$$\mathrm{dp}[i][j] = \max\begin{Bmatrix} \mathrm{dp}[i-1][j-1] + \mathrm{s}(r_i, a_j) \\ \mathrm{dp}[i-1][j] + g \\ \mathrm{dp}[i][j-1] + g \end{Bmatrix}$$

Where $s(r_i, a_j)$ denotes the score assigned to a pair of aligned words and g represents the gap penalty. The first row and the first column are initialized using cumulative gap penalties.

$$dp[i][0] = i \times g$$

$$dp[0][j] = j \times g$$

After the matrix has been filled, the optimal alignment is recovered by tracing back from the final matrix cell to the origin. This procedure guarantees the optimal global alignment under the selected scoring function. The algorithm requires (O(mn)) time and $O(mn)$ space.

B. Global Sequence Alignment for Subtitle Reconstruction

The Smith-Waterman algorithm performs local sequence alignment by identifying the highest-scoring matching region between two sequences. Unlike the Needleman-Wunsch algorithm, which aligns complete sequences, Smith-Waterman ignores regions outside the optimal local match.

This characteristic makes Smith-Waterman suitable for applications in which only partial similarity is required. Subtitle reconstruction, however, requires every reference word to be aligned with the corresponding ASR transcript. Since complete sequence correspondence must be preserved, global alignment provides a more appropriate formulation, making the Needleman-Wunsch algorithm the preferred choice for this work.

C. Text Alignment for Speech Applications

Text alignment has been widely used in Automatic Speech Recognition (ASR) applications, including broadcast news transcription, lecture transcription, and speech evaluation. These methods generally apply global sequence alignment or edit-distance algorithms after transcription has been completed.

Most existing approaches assume that sentence boundaries are already available or perform alignment independently for each sentence. Such assumptions simplify the alignment process but become less practical when multiple consecutive sentences must be aligned before sentence boundaries have been identified. In these situations, alignment errors may propagate across neighboring sentences, reducing the accuracy of subtitle reconstruction

D. Subtitle and Caption Alignment

Subtitle alignment has traditionally relied on acoustic information extracted from the original audio signal. Previous studies have employed forced alignment techniques with acoustic models and audio-driven synchronization based on Hidden Markov Models (HMMs). These methods estimate subtitle timing directly from speech and generally require access to both the audio recording and the acoustic model used during transcription.

In many practical applications, only the ASR transcript with word-level timestamps and the reference script are available. A text-based alignment approach is therefore more practical because it eliminates the dependency on the original audio while preserving the information required for subtitle reconstruction and timing synchronization

E. Context-Sensitive and Affine Gap Penalties

Context-dependent scoring has long been incorporated into dynamic programming algorithms to improve alignment quality. One of the most widely used approaches is Gotoh's affine gap penalty model, which assigns different penalties to opening a gap and extending an existing gap. This modification reduces unnecessary gap fragmentation while preserving the dynamic programming framework.

The proposed Segment-Aware Dynamic Programming (SADP) algorithm adopts the same principle of context-dependent scoring. Instead of modifying gap penalties, SADP introduces a boundary-aware penalty that is applied when an alignment transition crosses a sentence boundary detected in only one of the two aligned sequences. This additional constraint discourages incorrect cross-boundary matches while preserving the original recurrence relation and the asymptotic time and space complexity of the Needleman-Wunsch algorithm.

III. Research Methodology

A. Problem Formulation

The reference script consists of (K) sentences and is represented as a concatenated sequence of words $(R = (r_1, r_2, \ldots, r_m))$, where (r_i) denotes the (i)-th word in the reference script. Likewise, the Automatic Speech Recognition (ASR) transcript is represented as $(A = (a_1, a_2, \ldots, a_n))$, where each ASR word (a_j) is associated with its corresponding timestamp. In addition, every ASR word carries a provenance label (src(a_j)), which identifies the sentence from which the word originally came before artificial noise injection. These provenance labels are used solely for evaluation and are not available to the alignment algorithm during alignment.

The reference boundary set, denoted by (B_R), contains index (k) whenever word (r_k) is the final word of a sentence, that is, $(k \in B_R)$ if and only if $(r_k)$is immediately followed by sentence-ending punctuation. Similarly, the acoustic boundary

set (B_A)contains index (k)whenever the silence between two consecutive ASR words satisfies $(gap(a_k, a_{k+1}) \geq \delta)$, where $(\delta = 250)$ms. The objective is to determine an alignment that maximizes the total word-matching score while minimizing incorrect alignments that cross sentence boundaries.

Both (B_R) and $(B_A)$are implemented as Python hash sets, enabling average-case constant-time membership testing, $\big(O(1)\big)$,during each dynamic programming (DP) transition. Consequently, sentence-boundary information can be incorporated with negligible computational overhead.

To evaluate the contribution of each boundary source, two ablation settings are introduced. In the **Punctuation Missing** setting, all sentence-ending punctuation is removed from the reference script, resulting in $(B_R = \emptyset)$. In the **Pause Missing** setting, all inter-sentence pauses are compressed to less than 50 ms, resulting in $(B_A = \emptyset)$. These modifications affect only the inputs to the alignment algorithm; the ground-truth provenance labels and evaluation boundaries remain unchanged throughout the experiments.

B. Boundary Detection

The The proposed Sentence-Aware Dynamic Programming (SADP) algorithm incorporates two complementary sources of sentence-boundary information: textual boundaries extracted from the reference script and acoustic boundaries detected from the ASR transcript.

Reference sentence boundaries are extracted deterministically from sentence-ending punctuation. Whenever a word is immediately followed by a period, question mark, or exclamation mark, its index is inserted into the reference boundary set (B_R). Since punctuation is explicitly available in the reference script, this extraction process is deterministic and requires only a single pass through the text.

Acoustic sentence boundaries are identified using pauses between consecutive ASR words. Let $\big(t_k^{end}\big)$denote the ending timestamp of word (a_k) and (t_{k+1}^{start})denote the starting timestamp of the following word (a_{k+1}). The silence duration between two consecutive words is computed as

$\big[gap(a_k, a_{k+1}) = t_{k+1}^{start} - t_k^{end}.\big]$An acoustic boundary is detected whenever the pause duration satisfies

$[gap(a_k, a_{k+1}) \geq 250 \text{ ms}.]$The threshold of 250 ms was selected because pauses of this duration generally correspond to natural sentence transitions in conversational speech while remaining robust to shorter hesitations that frequently occur within sentences.

Both boundary sets are stored as Python hash sets, allowing average-case constant-time membership testing, $\big(O(1)\big)$, during every dynamic programming transition. As a result, incorporating sentence-boundary information introduces only a negligible computational overhead compared with the standard Needleman–Wunsch algorithm.

C. Sentence-Aware Dynamic Programming (SADP) Algorithm

The proposed Sentence-Aware Dynamic Programming (SADP) algorithm extends the classical Needleman–Wunsch (NW) algorithm by introducing a penalty whenever an alignment transition crosses a sentence boundary on only one side. The intuition behind this approach is that words belonging to different sentences should not be aligned unless both the reference script and the ASR transcript indicate that the transition occurs at the same sentence boundary.

When both the reference script and the ASR transcript contain a boundary at the corresponding positions, the transition is considered structurally consistent and no additional penalty is applied. Likewise, if neither sequence contains a boundary, the transition remains within the same sentence and is treated as a normal alignment. However, when a sentence boundary exists in only one sequence, the transition incorrectly aligns words across different sentences. This inconsistency is detected using the exclusive OR (XOR) operator

$[cross(i, j, d) = ref_{cross}(i, d) \oplus asr_{cross}(j, d),]$ where (d) denotes the transition direction in the dynamic programming matrix.

For a diagonal transition, $(ref_{cross} = (i-2) \in B_R) and (asr_{cross} = (j-2) \in B_A)$. For an upward transition, which corresponds to inserting a gap in the ASR sequence, $(ref_{cross} = False) and (asr_{cross} = (j-2) \in B_A)$. For a leftward transition, which inserts a gap in the reference sequence, $(ref_{cross} = (i-2) \in B_R)and(asr_{cross} = False)$. The subtraction of two from the indicades accounts for the one-based indexing used in the dynamic programming matrix while mapping the indices back to the original zero-based word positions.

The classical Needleman–Wunsch recurrence is modified by incorporating the cross-boundary penalty into each candidate transition. The diagonal transition is computed as

$\big[d = dp[i-1][j-1] + score\big(r_i, a_j\big) + \Phi \cdot cross(i, j, \text{DIAG}),\big]$the upward transition as

$[u = dp[i-1][j] + GAP + \Phi \cdot cross(i, j, \text{UP}),]$and the leftward transition as

$[l = dp[i][j-1] + GAP + \Phi \cdot cross(i, j, \text{LEFT}).]$The dynamic programming value for each cell is then obtained by

$[dp[i][j] = \max(d, u, l).]$ Throughout this study, the scoring parameters are fixed to $(MATCH = +2)$, $(MISMATCH = -1)$, $(GAP = -1)$,and $(\Phi = -20)$. The boundary penalty (\Phi) was determined through a grid search on a validation set consisting of 30 passages, with candidate values ranging from −4 to −30. The selected value is sufficiently negative to discourage incorrect cross-boundary matches while avoiding excessively large penalties that would encourage long sequences of gaps.

The summarize of SADP algorithm :

```
MATCH=2; MISMATCH=-1; GAP=-1
_DIAG, _UP, _LEFT = 0, 1, 2
```

```
def _crosses(i, j, d, Br, Ba):
    rc = False if d==_UP   else (i-2) in Br
    ac = False if d==_LEFT else (j-2) in Ba
    return rc ^ ac

def sadp_align(R, A, Br, Ba, phi=-20):
    m, n = len(R), len(A)
    dp=[[0.0]*(n+1) for _ in range(m+1)]
    tb=[[_DIAG]*(n+1) for _ in range(m+1)]
    for i in range(1, m+1):
        dp[i][0]=i*GAP; tb[i][0]=_UP
    for j in range(1, n+1):
        dp[0][j]=j*GAP; tb[0][j]=_LEFT
    for i in range(1, m+1):
      for j in range(1, n+1):
        s=MATCH if R[i-1]==A[j-1] else MISMATCH
        d_sc=dp[i-1][j-1]+s
        u_sc=dp[i-1][j]+GAP
        l_sc=dp[i][j-1]+GAP
        if _crosses(i,j,_DIAG,Br,Ba):
            d_sc += phi
        if _crosses(i,j,_UP,  Br,Ba):
            u_sc += phi
        if _crosses(i,j,_LEFT,Br,Ba):
            l_sc += phi
        if d_sc>=u_sc and d_sc>=l_sc:
            dp[i][j]=d_sc; tb[i][j]=_DIAG
        elif u_sc>=l_sc:
            dp[i][j]=u_sc; tb[i][j]=_UP
        else:
            dp[i][j]=l_sc; tb[i][j]=_LEFT
    aln,i,j=[],m,n
    while i>0 or j>0:
        if i>0 and j>0 and tb[i][j]==_DIAG:
            aln.append((R[i-1],A[j-1]))
            i-=1; j-=1
        elif i>0:
            aln.append((R[i-1],None)); i-=1
        else:
            aln.append((None,A[j-1])); j-=1
    return aln[::-1], dp[m][n]
```

D. Complexity Analysis

The doubly-nested loop runs for $m \times n$ iterations. Each iteration performs three hash-set membership tests, each $O(1)$amortised for Python sets, plus constant arithmetic and a three-way max. The traceback traverses at most m + n steps. Total time complexity is $O(m \times n)$,identical to standard NW. Space is $O(m \times n)$for the dp and tb matrices, reducible to $O(min(m,n))$if only the score is required and traceback is omitted.

In practical terms, the hash-set lookups add a small constant overhead per cell (approximately 5–8% in Python benchmarks for passage lengths of 20–150 words). For very long passages, the $O(m \times n)$space requirement remains the binding constraint. The boundary sets B_R and $B_A$each require $O(K)$space, where K is the number of sentence boundaries, negligible compared to the DP matrix.

IV. Experimental Setup

A. Dataset

Experiments use the Reuters-21578 News Corpus, a standard benchmark available through the NLTK library, requiring no API key or commercial license. Articles span 10 financial topic categories: trade, earnings, acquisitions, money-fx, interest, shipping, grain, crude, employment, and GNP. Reuters was chosen because its articles exhibit strong intra-article vocabulary overlap: news about a corporate acquisition repeatedly uses words such as "shares", "company", "acquisition", and "dlrs" across consecutive sentences. This creates a realistic cross-boundary alignment challenge in which the NW aligner has a genuine incentive to match words from the wrong sentence.

TABLE I. DATASET STATISTICS

Parameter	Value
Total sentences	400
Source articles	58
Passages (article groups)	58
Sentences per passage	2-10
Sentence length	5-30 words
Inter-word gap	50-100 ms (synthetic)
Inter-sentence pause	300-500 ms (synthetic)

Articles with fewer than four usable sentences (5-30 words each, after filtering headers and short noise lines) are discarded. Up to ten sentences per article are retained and concatenated into a single passage, yielding 58 passages from 58 articles (6.9 sentences on average). To stress-test boundary detection, deletion probability is boosted 2.5x at sentence-boundary word positions during noise injection. Word-level timestamps are generated synthetically: each word duration is estimated at 80 ms per character (minimum 200 ms per word), inter-word gaps are drawn uniformly from [50, 100] ms, and inter-sentence pauses are drawn uniformly from [300, 500] ms, placing all inter-sentence gaps well above the 250 ms detection threshold under the control condition.

B. Noise Injection

Three noise types are injected at the word level, each with independent probability p = noise_level / 3. A homophone substitution map of 60+ pairs (e.g., "their" / "there", "its" / "it's", "by" / "buy", "hear" / "here") represents phonetically confusable word pairs common in English ASR. Single-character edit substitutions change one interior character of the word (cycling through the alphabet), simulating near-miss transcription errors. Filler-word insertions ("um", "uh", "like", "basically", "so") inherit the provenance label of the preceding word, preserving ground-truth evaluation integrity. All noise decisions are made independently per word; sentence boundaries as defined by the reference text are not modified during noise injection.

TABLE II. NOISE TYPES AND MECHANISMS

Noise Type	Mechanism
Deletion	Word removed with probability p = noise/3
Substitution	Replaced with homophone or single-character edit
Insertion	Filler word ("um","uh","like") inserted after word

Three total noise levels are tested: 10% (p = 0.033 per type), 20% (p = 0.067), and 35% (p = 0.117). For the ablation

study, two additional boundary-corruption modes are applied independently at 20% noise: Punct Missing strips all sentence-ending punctuation from the reference text before passing it to the algorithm (setting $B_R = empty$); Pause Missing compresses all inter-sentence acoustic gaps to a random value in [20, 50] ms before boundary detection (setting $B_A = empty$). These corruptions affect only algorithm inputs; ground-truth provenance labels and evaluation boundaries are unchanged.

C. Evaluation Matrix

(i, aj) in the alignment output where neither element is a gap, WAA checks whether the reference sentence containing r*i* equals src(a*j*). WAA = (correct pairs) / (total non-gap pairs). The metric is independent of transcription accuracy; it measures only whether the alignment placed the ASR word in the correct reference sentence, which is the central objective of SADP.

Boundary F1 measures the spatial accuracy of sentence boundary prediction in the ASR word-index space. Ground-truth boundaries are positions where the provenance tag changes in the ASR word sequence. Predicted boundaries are positions where the alignment path transitions between reference sentences. A predicted boundary is counted as a true positive if it falls within plus or minus 2 word positions of a ground-truth boundary. Precision, recall, and F1 are computed in the standard way.

V. Results

A. Experiment A: Main Comparison

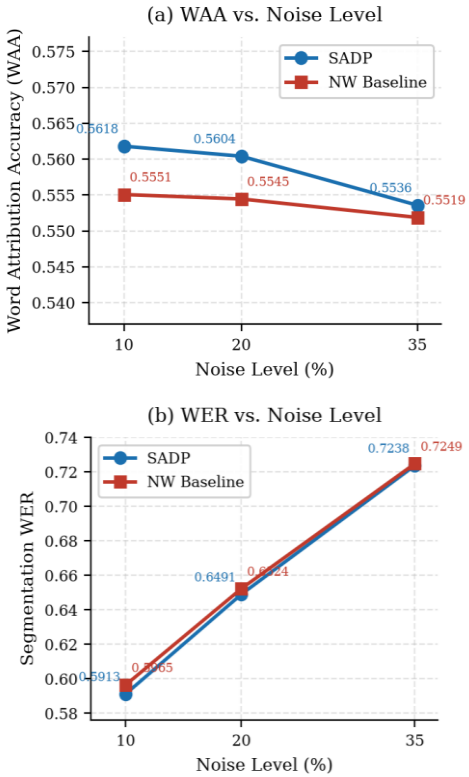
Table III reports the mean WAA, Seg-WER, and Boundary F1 averaged across all 58 passages for each noise level and method. Table IV summarises the relative improvement of SADP over NW. Figure 1 shows WAA and Seg-WER as a function of noise level.

TABLE III. SADP VS. NW BASELINE — MAIN COMPARISON

Noise	Method	WAA	Seg-WER	F1 Bound.
10%	**SADP**	0.5618	0.5913	0.8291
10%	NW Baseline	0.5551	0.5965	0.8303
20%	**SADP**	0.5604	0.6491	0.8272
20%	NW Baseline	0.5545	0.6524	0.8303
35%	**SADP**	0.5536	0.7238	0.8238
35%	NW Baseline	0.5519	0.7249	0.8303

TABLE IV. RELATIVE IMPROVEMENT OF SADP OVER NW

Noise	WAA Improvement	Seg-WER Reduction
10%	+1.21%	-0.88%
20%	+1.07%	-0.50%
35%	+0.31%	-0.15%



SADP outperforms NW on both WAA and Seg-WER at all three noise levels. The WAA improvement ranges from 0.31% at 35% noise to 1.21% at 10% noise; the Seg-WER reduction from 0.15% to 0.88%. The declining margin is consistent with theory: at higher noise, boundary words are more frequently deleted (2.5x boundary-boost probability), reducing the informativeness of both signals. Boundary F1 scores are nearly identical (0.769-0.772 for SADP vs. 0.772 for NW), as minor path differences near boundaries shift predicted ASR boundary positions by only 1-2 words. Both methods show monotonically increasing Seg-WER (0.57 to 0.70), confirming that the noise injection is effective and that SADP's advantage persists across all conditions.

References

[1] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," J. Mol. Biol., vol. 48, no. 3, pp. 443-453, 1970. [Online]. Available: https://doi.org/10.1016/0022-2836(70)90057-4

[2] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," J. Mol. Biol., vol. 147, no. 1, pp. 195-197, 1981. [Online]. Available: https://doi.org/10.1016/0022-2836(81)90087-5

[3] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in Proc. ICML, Honolulu, HI, 2023, pp. 28492-28518. [Online]. Available: https://arxiv.org/abs/2212.04356

[4] M. Bain, J. Huh, T. Han, and A. Zisserman, "WhisperX: Time-accurate speech transcription of long-form audio," arXiv preprint arXiv:2303.00747, 2023. [Online]. Available: https://arxiv.org/abs/2303.00747

[5] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal Forced Aligner: Trainable text-speech alignment using Kaldi,"

in Proc. Interspeech, Stockholm, 2017, pp. 498-502. [Online]. Available: https://doi.org/10.21437/Interspeech.2017-1386

[6] P. Lison and J. Tiedemann, "OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles," in Proc. LREC, Portoroz, 2016, pp. 923-929. [Online]. Available: https://aclanthology.org/L16-1147

[7] O. Gotoh, "An improved algorithm for matching biological sequences," J. Mol. Biol., vol. 162, no. 3, pp. 705-708, 1982. [Online]. Available: https://doi.org/10.1016/0022-2836(82)90398-9

[8] D. D. Lewis, "Reuters-21578 text categorization test collection, distribution 1.0," AT&T Labs-Research, 1997. [Online]. Available: https://archive.ics.uci.edu/dataset/137/reuters-21578+text+categorization+collection

ACKNOWLEDGMENT

I gratefully acknowledges the blessings and strength granted by God Almighty, which enabled the successful completion of this paper. Sincere appreciation is also extended to Dr. Ir. Rinaldi, M.T., lecturer of the IF2211Algorithmic Strategies, for his continuous support, guidance, and encouragement throughout the semester and in the writing of this paper. Next, I would thank both of my parents and my girlfriend who always give the author supports during ups and downs during the process of this work.

APPENDIX

Github : https://github.com/Sakazu01/Makalah_Stima

STATEMENT

Hereby I declare that this paper that I have written is my own work, not a reproduction or translation of someone else's work and not plagiarized.

Bandung, 25 Juni 2026

Ariel Cornelius Sitorus - 13524085