Actes de CORIA-TALN 2026
Nantes, France, 29 juin au 3 juillet 2026

Raffinement itératif pour l'analyse des dépendances temporelles à l'échelle du document

Ensieh Hemmatan, Timothée Bernard, Benoît Crabbé
Université Paris Cité, CNRS, LLF, Paris, France
ensieh.hemmatan-attarbashi@etu.u-paris.fr
timothee.bernard@u-paris.fr, benoit.crabbe@u-paris.fr

RÉSUMÉ

L'analyse de dépendances temporelles vise à prédire des dépendances entre événements et expressions temporelles. Les modèles neuronaux traditionnels s'appuient sur des encodeurs préentraînés et un modèle de scorage des relations possibles. Les relations sont prédites indépendamment les unes des autres sans modéliser explicitement leurs inter-dépendances. De manière à approcher une prédiction globale, nous proposons un analyseur supervisé qui alterne prédiction de graphe et raffinement des représentations des nœuds. Notre modèle itératif obtient des gains, par rapport à notre modèle de base, de +2.38 points en macro F1 étiqueté (au niveau du document) et de +2.09 points en macro F1 non étiqueté. Nos analyses sur l'ensemble de développement montrent des gains monotones et une augmentation de la stabilité des prédictions au fil des étapes de raffinement. Ces résultats montrent l'intérêt du raffinement itératif comme méthode de modélisation pour l'analyse de dépendances temporelles à l'échelle du document.

ABSTRACT

Iterative Refinement for Document-Level Temporal Dependency Parsing

Temporal Dependency Parsing (TDP) aims to predict directed temporal dependencies between events and time expressions. Typical neural dependency parsing systems use pretrained encoders with edge-factored scoring and predict all edges in one pass, which might limit explicit structure modeling. We propose a supervised temporal dependency parser that alternates graph prediction and graph neural network message passing to refine node representations. The selected iterative model with three iterations (T=3) improves over a matched baseline (T=0) by +2.38 points in labeled document-level macro F1 and +2.09 points in unlabeled document-level macro F1 on the test set. Per-iteration diagnostics on the development set show monotonic gains and increasing stability across refinement steps. These results support iterative refinement as a useful modeling choice for document-level temporal dependency parsing.

MOTS-CLÉS : analyse de dépendances temporelles, raffinement itératif, graphes de dépendances temporelles, réseaux de neurones sur graphes.

KEYWORDS: Temporal Dependency Parsing, Iterative Refinement, Temporal Dependency Graph, Graph Neural Network.

Actes des 33ème Conférence sur le Traitement Automatique des Langues Naturelles.
Volume 1 : articles scientifiques originaux, pages 470–482.

1 Introduction

Understanding time and temporal structure is fundamental to reasoning about world dynamics in text (Pustejovsky *et al.*, 2004) and is also vital for improving downstream applications such as question answering and clinical diagnostic models (Huang *et al.*, 2023; Chaturvedi *et al.*, 2025; Pustejovsky *et al.*, 2004). A common approach to the problem is to predict a temporal relation for each candidate pair of nodes, each node representing an event mention (event node) or a time expression (timex node) in text (Cassidy *et al.*, 2014). This pairwise view scales quadratically with the number of temporal mentions, can make globally coherent decoding harder, and can also lead to inconsistent annotations (Zhang & Xue, 2018). Alternative formalisms reduce this burden by using temporal dependency representations instead (Bethard *et al.*, 2012; Zhang & Xue, 2018; Yao *et al.*, 2020).

Zhang & Xue (2018) introduced the Temporal Dependency Tree (TDT) representation, in which event and timex nodes, and a small set of meta nodes form a rooted edge-labeled structure. In this representation, each node attaches to a single parent that serves as a temporal reference, that is, the antecedent that most precisely determines its temporal location. This tree-based formulation reduces annotation complexity and allows additional temporal relations to be inferred transitively. Yao *et al.* (2020) later extended the TDT representation to Temporal Dependency Graphs (TDG), arguing that the single-parent assumption is sometimes too restrictive. In a TDG, each event has an obligatory reference timex and may additionally take a reference event. These representations are therefore more expressive for document-level temporal reasoning while retaining a structured prediction target.

Recent neural work has improved temporal dependency parsing by strengthening document encoders and graph modeling. Ross *et al.* (2020) study contextualized neural language models for temporal dependency tree parsing and show that BERT (Devlin *et al.*, 2019) substantially improves over earlier BiLSTM-based systems. Mathur *et al.* (2022) propose DocTime, a document-level TDG parser that combines contextual representations with richer graph encodings and auxiliary path-prediction supervision to capture longer-range dependencies. More recently, Yao *et al.* (2023) propose an NLI-based TDG parser that turns candidate labeled TDG edges into natural language statements and uses a pretrained entailment model to score them.

Prior TDG parsers improve encoders, graph features, and task formulations, but do not isolate iterative graph refinement as a controlled design choice. Although DocTime (Mathur *et al.*, 2022) includes an iterative graph-learning component within a larger graph-based parser, it does not compare a matched single-pass parser with iterative variants or analyze how predictions evolve across refinement steps.

Temporal-relation work has addressed global coherence with ILP-based constrained inference and, in later work, endpoint inference and graph decomposition (Chambers & Jurafsky, 2008; Denis & Muller, 2011). Rather than imposing such constraints, we ask whether message passing over predicted temporal graphs can refine node representations. More broadly, graph-based parsing has explored multi-pass coarse-to-fine decoding, graph-to-graph refinement, and arc-vector refinement architectures (Rush & Petrov, 2012; Mohammadshahi & Henderson, 2021; Floquet *et al.*, 2025).

Work on iterative graph learning outside temporal parsing provides a natural motivation for studying iterative refinement in TDG parsing. Chen *et al.* (2020) proposed Iterative Deep Graph Learning (IDGL), which alternates graph induction and graph neural network (GNN)-based representation learning, exploiting the feedback loop in which better node representations yield better graphs and better graphs in turn improve the representations. In parallel, edge-factored graph prediction with biaffine scoring has become a standard design in dependency parsing (Dozat & Manning, 2018).

Taken together, these lines of work suggest a design for TDG parsing : predict a graph from the current node representations, update those representations with message passing over the predicted graph, and then rescore the graph from the refined representations.

To test whether iterative graph refinement improves TDG parsing, we introduce a supervised parser that alternates between graph prediction and Graph Neural Network (GNN)-based representation updates. Starting from contextual BERT representations, our model predicts dense edge-existence scores, converts them into a sparse graph through learned sparsification and exponential moving average (EMA) blending, updates node representations with a GNN, and repeats this predict-update cycle for a fixed number of refinement steps. Edge labels are scored from the resulting node states. This lets us test a focused hypothesis : whether iterative refinement improves document-level TDG parsing over a matched single-pass baseline. For final evaluation, edge existence is decoded with a single development-calibrated global threshold per run, applied to the sigmoid-transformed logits from the final scored state ; the same protocol is used for both $T{=}0$ and $T{>}0$.

Contributions. In this paper, we introduce a document-level temporal dependency parser that alternates graph prediction and GNN-based representation updates, allowing temporal graphs to be rescored from progressively refined representations. Empirically, the selected iterative model with three refinement steps improves over the single-pass baseline by 2.38 points in labeled and 2.09 points in unlabeled document-level macro F1 on the test set. Finally, we provide per-iteration analyses showing monotonic gains and increasing prediction stability, which support the choice of refinement steps and clarify how the iterative procedure converges.

2 Task and Model

2.1 Temporal Dependency Graph Parsing

Given a document and its pre-identified temporal nodes (gold event and timex spans), the task is to predict a temporal dependency graph. Figure 1 shows a concrete example from a short training document. In the TDG format, a directed edge points from a child node to the parent relative to which it is temporally interpreted. Edges are labeled with DEPEND-ON, INCLUDED, BEFORE, AFTER, and OVERLAP. Following Yao *et al.* (2020) and the dataset, BEFORE, AFTER, and OVERLAP describe temporal ordering or overlap between a node and its reference parent. INCLUDED marks inclusion in the reference time, and DEPEND-ON is used for structural attachments, including event attachment to ROOT when no reference event is annotated, and timex anchoring to the document creation time (DCT).

2.2 Model

The model is a biaffine edge-factored parser that iteratively refines the predicted temporal graph. First, the text is encoded by a BERT encoder to produce initial node representations from event and timex spans. Biaffine scorers predict dense adjacency and edge-label scores, and a GNN updates node features between successive graph predictions when $T{>}0$. The model predicts edge scores at every state $t \in \{0, \ldots, T\}$, yielding $T{+}1$ scored states in total. $t{=}T$ is a final rescoring step and the single-pass baseline is obtained by setting $T{=}0$.

August 22, 2005 (t_0)

In San Pasqual a brush fire (e_1) was dealt (e_2) with as it burned (e_3) four acres of wildlands. This may have been due to sparks (e_4) from a vehicle accident (e_5). The blaze lasted (e_6) from 3 :30 (t_1) to 4 :30 PM (t_2) until fire (e_7) trucks and many other emergency vehicles helped (e_8) stop (e_9) the blaze.

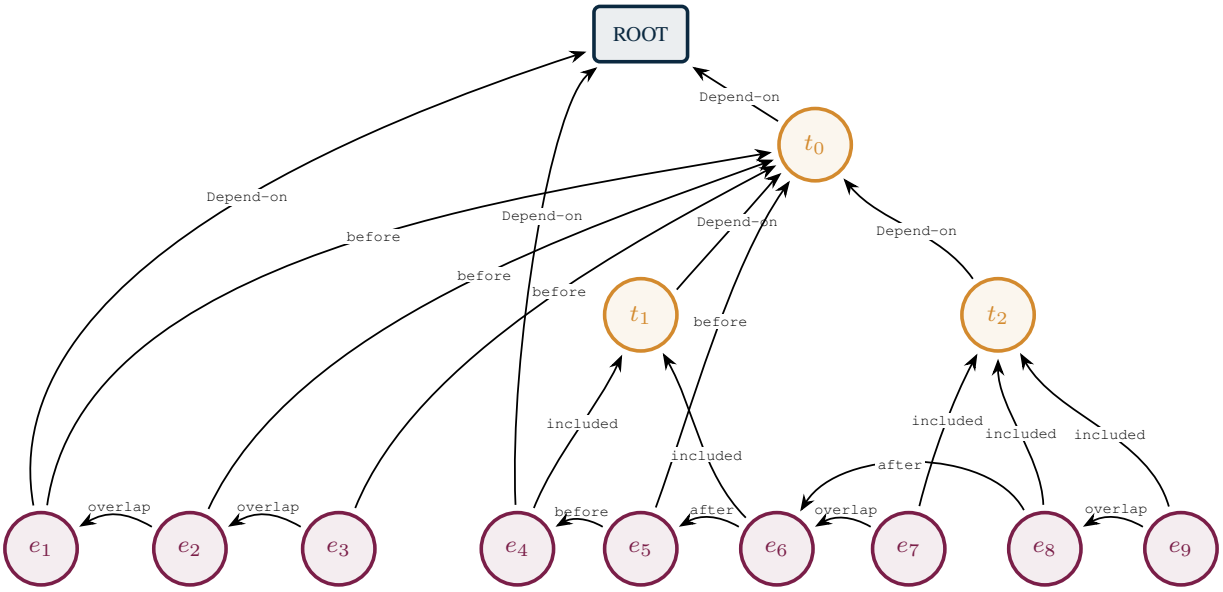

FIGURE 1 – A TDG example from a training document. Underlined spans are event and timex nodes; edges point from a child node to its reference parent and are labeled with the dependency type.

2.2.1 Problem Formulation

Given n nodes (event or timex) $\mathcal{V} = \{1, \ldots, n\}$, our goal is to predict a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each edge is labeled with a temporal dependency type $\ell \in \mathcal{L}$. Let x denote the input document and $y^\star \in \{0,1\}^{n \times n}$ the gold adjacency matrix. A pretrained encoder Enc_θ, fine-tuned during parser training, maps the document x to contextual subword representations, from which we construct the initial node representations $\mathbf{H}^{(0)} = [\mathbf{h}_1^{(0)}, \ldots, \mathbf{h}_n^{(0)}] \in \mathbb{R}^{n \times d_0}$ as described below. We use BERT-base-cased in our experiments.

For encoding long documents, we use overlapping sliding windows and average overlapping subword states. For each event or timex span, we mean-pool the contextual subword states covered by that span to obtain a node encoding. We then concatenate this vector with learned embeddings for node type, node position in the document, and sentence index. The resulting vectors form the initial node matrix $\mathbf{H}^{(0)}$ used by the graph scorer and by the refinement loop. We write $\mathbf{H}^{(t)} \in \mathbb{R}^{n \times d_t}$ for later refinement states, since these encoder-based node vectors are projected to the GNN hidden size before message passing. For $t > 0$, $\mathbf{H}^{(t)}$ denotes node representations after t graph-conditioned refinement steps.

2.2.2 Temporal Graph Prediction

The parser predicts both edge existence and edge labels with biaffine classifiers. Following the biaffine parsing design of Dozat & Manning (2018), at each scoring step we map the node representations

used for that decision to role-specific source and target projections. This lets the model encode the fact that a node may behave differently when it is scored as a reference parent versus as a child node.

We then score every node pair with two biaffine classifiers. The edge classifier produces a scalar edge logit $z_{ij}^{(t)}$ from the current graph-scoring state, and collecting these scores over all pairs gives the dense adjacency matrix $\mathbf{Z}^{(t)} \in \mathbb{R}^{n \times n}$. The label classifier produces a label-score vector $\mathbf{e}_{ij}^{(t)} \in \mathbb{R}^{|\mathcal{L}|}$ for each candidate edge from the node state used for label prediction at that step. For non-final refinement steps, this label-scoring state is the post-message-passing state; at the final step, it is $\mathbf{H}^{(T)}$. This edge-factored formulation keeps structure prediction and label prediction closely coupled while remaining simple to compute over all node pairs.

This choice is motivated by two properties emphasized by Dozat & Manning (2018) : biaffine scorers are expressive enough to model pairwise interactions between role-specific token representations, and they work well in graph-based parsing without requiring heavy global inference. In our setting, the same scoring mechanism is reused at every iteration, so any change in the predicted graph comes only from the refined node representations. Final decoding uses the label scores from the last scored state.

2.2.3 Iterative Graph Refinement

Iterative refinement is the central modeling idea of our parser. The motivation is close in spirit to Chen *et al.* (2020) : better node representations can support better graph estimates, and better graph estimates can in turn improve the node representations. Our setting is nevertheless more specific than IDGL. We do not start from a noisy observed graph and learn a general similarity metric for node classification. Instead, at each iteration we score temporal edges from the current node representations, convert these scores into a sparse message-passing graph, and then rescore the graph from the updated representations. The aim here is controlled iterative refinement for supervised temporal dependency parsing.

At each state $t \in \{0, \ldots, T\}$, the model recomputes dense edge-existence logits from $\mathbf{H}^{(t)}$. If $t < T$, these scores are converted into a sparse message-passing graph and used to produce $\mathbf{H}^{(t+1)}$. The label scores associated with this intermediate step are computed after this update, so they supervise the post-update representation. At $t = T$, no further message passing is applied; final edge existence and final edge labels are both decoded from $\mathbf{H}^{(T)}$. This loop gives the model an explicit mechanism for revising earlier edge hypotheses after information has been propagated through the current predicted graph.

Sparsification. We convert raw logits $\mathbf{Z}^{(t)}$ to sparse probability matrices using a differentiable Gumbel–Sigmoid sparsifier (Jang *et al.*, 2017) with a learnable threshold $\tau \in (0, 1)$:

$$p_{ij}^{(t)} = \sigma\left(\frac{z_{ij}^{(t)} - \operatorname{logit}(\tau) + g_{ij}}{\kappa}\right) \tag{1}$$

where σ is the sigmoid function, $g_{ij} \sim \mathrm{Gumbel}(0, 1)$ is Gumbel noise (training only), and κ is the temperature. This yields a sparse probability matrix $\mathbf{P}^{(t)} \in [0, 1]^{n \times n}$ with self-loops removed. At evaluation time, the stochastic noise is removed. In contrast to IDGL, which learns a graph through a similarity metric and neighborhood sparsification, our graph is induced directly from temporal edge logits. This keeps the refinement signal tied to the parsing objective instead of introducing a separate graph-learning module.

EMA blending. To stabilize updates across iterations, we blend the current sparse probabilities with the previous blended graph using Exponential Moving Average (EMA) :

$$\mathbf{Q}^{(t)} = \lambda \mathbf{Q}^{(t-1)} + (1-\lambda)\mathbf{P}^{(t)} \tag{2}$$

with $\mathbf{Q}^{(0)} = \mathbf{P}^{(0)}$. The scalar λ controls how strongly the previous graph is retained. IDGL combines learned graph structures with an initial observed graph, whereas we do not assume such a graph is available. Instead, we smooth only across successive predicted graphs, which reduces oscillation while still allowing the graph to evolve over iterations.

Normalization for message passing. During row normalization, zero-mass rows are repaired with a self-loop row. We then mix the normalized matrix with identity :

$$\operatorname{norm}\big(\mathbf{Q}^{(t)}\big)[i,\cdot] = \begin{cases} \dfrac{\mathbf{Q}^{(t)}[i,\cdot]}{\sum_j q_{ij}^{(t)}}, & \text{if } \sum_j q_{ij}^{(t)} > 0, \\ \mathbf{u}_i^{\top}, & \text{otherwise,} \end{cases} \tag{3}$$

$$\hat{\mathbf{A}}^{(t)} = (1-\alpha)\operatorname{norm}\big(\mathbf{Q}^{(t)}\big) + \alpha\,\mathbf{I} \tag{4}$$

where $\mathbf{u}_i$ is the i-th standard basis vector in $\mathbb{R}^n$ and α is a tunable hyperparameter. This guarantees that each node receives self-information during message passing. The normalization also turns the blended graph into a stable carrier of relational information for the next update step.

Message passing and rescoring. For each $t < T$, we apply L layers of Graph Convolutional Networks (GCN) to update node representations :

$$\mathbf{H}^{(t+1)} = \mathrm{GNN}\left(\mathbf{H}^{(t)}, \hat{\mathbf{A}}^{(t)}\right) \tag{5}$$

At the last scored state $t{=}T$, we do not apply an additional GNN update ; final decoding uses $\mathbf{H}^{(T)}$.

Decoding. Edge existence is determined by applying one global threshold $\tau^{\star}$, calibrated on the development set, to the sigmoid-transformed dense scores from the final scored state. Edge labels are assigned with $\arg\max_{\ell} \mathbf{e}_{ij,\ell}^{(T)}$. This rule is used only for final evaluation ; the learned sparsifier is used earlier in the forward pass to build the message-passing graph during refinement. No global decoding constraints are imposed.

2.2.4 Training Objectives

We supervise edge existence at every graph-scoring state and edge labels at the corresponding label-scoring outputs, rather than only at the end of the refinement loop. This choice encourages the model to produce usable intermediate graphs throughout the trajectory, while still emphasizing later states that are closer to the final decoded prediction. Concretely, all losses are aggregated with linearly increasing iteration weights.

Graph structure loss. At each scored state t, we apply binary cross-entropy with logits to the dense adjacency logits $\mathbf{Z}^{(t)}$ against the gold adjacency $y^{\star}$ over all node pairs. This loss gives direct supervision to the graph scorer at each refinement step, so the message-passing graph is always tied to the edge-existence objective rather than learned only indirectly through the final prediction.

We then aggregate graph losses with a weighted mean :

$$\mathcal{L}_{\text{graph}} = \sum_{t=0}^{T} w_t \, \ell_{\text{graph}}^{(t)}, \quad w_t \propto (t{+}1), \quad \sum_{t=0}^{T} w_t = 1 \tag{6}$$

The increasing weights bias training toward later scored states, which are closest to the model used at evaluation time, while still retaining stabilizing supervision on earlier states.

Edge labeling loss. We use focal loss (Lin *et al.*, 2020) with class balancing to handle the imbalanced relation distribution. Let $\mathcal{E}^\star = \{(i,j) : y_{ij}^\star = 1\}$ denote the gold edge set, and let $\ell_{ij}^\star$ be the gold label for edge (i,j) :

$$\ell_{\text{label}}^{(t)} = -\frac{1}{|\mathcal{E}^\star|} \sum_{(i,j)\in\mathcal{E}^\star} \sum_{\ell\in\mathcal{L}} \alpha_\ell (1 - \pi_{ij,\ell}^{(t)})^\gamma \, \mathbf{1}\{\ell_{ij}^\star = \ell\} \log \pi_{ij,\ell}^{(t)} \tag{7}$$

Here, $\pi_{ij,\ell}^{(t)} = \mathrm{softmax}(\mathbf{e}_{ij}^{(t)})_\ell$, γ is the focusing parameter, and α_ℓ are class-balanced weights. We apply this loss only on gold edges, since relation labels are defined only when an edge exists. Similar to graph structure loss, we aggregate across iterations with weighted mean :

$$\mathcal{L}_{\text{label}} = \sum_{t=0}^{T} w_t \, \ell_{\text{label}}^{(t)} \tag{8}$$

Using the same iteration-weight schedule for both losses emphasizes later graph and label predictions consistently.

Overall objective. The final loss combines both objectives :

$$\mathcal{L} = \beta \mathcal{L}_{\text{graph}} + (1-\beta)\mathcal{L}_{\text{label}} \tag{9}$$

where $\beta \in [0,1]$ controls the relative weight of graph and label supervision. This balancing is important because the two objectives play different roles : $\mathcal{L}_{\text{graph}}$ shapes the structure used for message passing, while $\mathcal{L}_{\text{label}}$ determines how well the predicted edges are semantically typed.

2.2.5 Baseline Configuration

The baseline is designed as a matched control rather than as a separate parser. It shares the same Transformer encoder, feature augmentation, biaffine edge and label scorers, and training objectives as the iterative model. The only substantive difference is that it sets $T{=}0$, so the refinement loop is disabled : the model predicts edge-existence and label scores once from the initial node representations and does not apply any GNN-based message-passing update.

This distinction matters for interpreting the results. Because the single-pass baseline and the iterative model use the same encoder, scorers, supervision, and evaluation rule, the comparison focuses on the effect of graph-conditioned refinement updates.

At evaluation time, the baseline is decoded under the same protocol as the iterative model and scored with the same metrics. This makes $T{=}0$ a fair single-pass reference point for assessing whether repeated graph prediction and message passing improve temporal dependency parsing.

3 Experiments

Dataset. We evaluate on the TDG corpus of Yao *et al.* (2020), which contains 500 documents split into train, development, and test as 400, 50, and 50. As described in Section 2, documents are annotated with event and timex nodes and labeled temporal dependency edges. We assume gold node spans and evaluate only the dependency parsing stage. The label distribution is imbalanced, with BEFORE and DEPEND-ON among the most frequent relations and AFTER among the least frequent. This imbalance motivates the class-balanced focal loss used in training.

For documents longer than the encoder context window, we use overlapping sliding-window tokenization with a maximum subword length of 510 and a stride of 350, then aggregate the resulting subword states back to node representations.

Protocol and settings. We use a two-stage protocol. First, we select the configuration on the development set. Second, we freeze the selected setting and report final test-set performance over seeds $\{30, 31, 32\}$. Model selection maximizes labeled document-level macro F1 on the development set over all edges. For each run, we calibrate one global threshold $\tau^\star$ on the development set and apply it unchanged on the test set. The same fixed $\tau^\star$ is also used for the per-iteration development-set diagnostics reported later.

Within the narrowed search space, the selected iterative setting is $T{=}3$, and the matched baseline is $T{=}0$. Both settings share the same encoder, biaffine scorers, training objective, and decoding protocol; setting $T{=}0$ disables refinement updates. We report $T{=}3$ because it is the development-selected iterative setting under our selection protocol. The per-iteration diagnostics support this choice : gains shrink by the last refinement step, while labeled self-Jaccard rises from 0.8368 to 0.9168 to 0.9439, indicating that the predicted graph has largely stabilized by iteration 3.

The reported runs keep the remaining configuration fixed; for the iterative model, refinement uses EMA graph updates with previous-graph retention weight $\lambda = 0.3$, a 2-layer GCN with hidden size 512, and a Gumbel-based sparsifier. To support reproducibility, we report the exact evaluation protocol, selected settings, and final seeds in the paper. The code can be found at `https://github.com/hemmatan/tdp_iterative_refinement`.

4 Results and Discussion

We first analyze how the iterative model evolves across refinement steps on the development set. We then report final test-set results for the selected model and situate them with respect to earlier TDG systems. The primary metrics are document-level macro F1 for labeled and unlabeled parsing.

4.1 Per-Iteration Performance Progression

To understand where gains come from, we analyze the iterative model ($T{=}3$) across iterations on the development set, aggregated over 3 seeds, using each run's fixed global $\tau^\star$ calibrated on that same set.

Figure 2 shows monotonic mean gains from $i{=}0$ to $i{=}3$ in both metrics (labeled : $58.29 \rightarrow 58.60 \rightarrow 59.03 \rightarrow 59.24$; unlabeled : $66.64 \rightarrow 66.93 \rightarrow 67.22 \rightarrow 67.45$). By $i{=}3$, cumulative gains relative to $i{=}0$ reach $+0.95 \pm 0.28$ percentage points in the labeled setting and $+0.81 \pm 0.23$ points in the

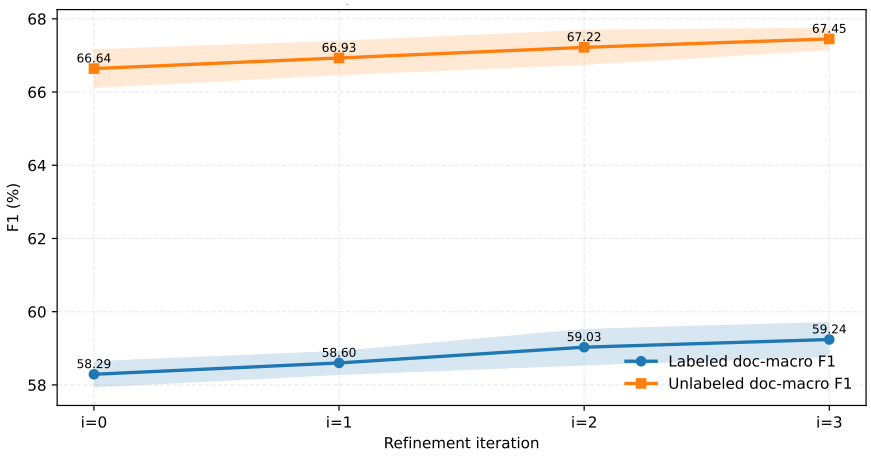

FIGURE 2 – Per-iteration progression on the development set for the iterative model (T=3), mean $\pm$ std over 3 seeds. Scores are document-level macro F1 using each run's fixed global $\tau^\star$ calibrated once on the development set.

unlabeled setting. The step from i=2 to i=3 remains positive but smaller than earlier gains, which suggests that the last refinement step mainly consolidates the graph rather than making large structural revisions.

This interpretation is supported by the stability analysis. We measure stability with labeled self-Jaccard, defined as the Jaccard similarity between the labeled predicted edge sets at two consecutive iterations ; higher values indicate that the graph changes less from one iteration to the next. Labeled self-Jaccard rises from 0.8368 to 0.9168 to 0.9439 over $(0\to1)$, $(1\to2)$, and $(2\to3)$, indicating that consecutive predicted graphs become increasingly similar as refinement proceeds. Taken together, the mean F1 curves and the self-Jaccard trend are consistent with a process in which early iterations make broader structural corrections and later iterations focus on smaller adjustments to an already stable graph.

4.2 Where Iterative Refinement Helps

To interpret these aggregate gains more closely, we examine development-set slice analyses from the iterative runs, including distance-based and relation-based breakdowns. The results suggest that the benefit of iteration comes primarily from representation updates conditioned on provisional graph structure, not from repeatedly applying a different final decoding rule. The single-pass baseline and the iterative model use the same scorer family and the same decoding protocol at evaluation time, so the main difference is whether node representations are revised through graph-conditioned message passing before the final scored state. The monotonic per-iteration curves and rising self-Jaccard values are consistent with this interpretation : early iterations make larger structural changes, while later iterations mostly consolidate an increasingly stable graph.

The slice analyses also suggest that the benefit of refinement is not uniform across structural conditions. From i=0 to i=3, same-sentence edges gain the most on average (+2.30 labeled points and +1.63 unlabeled points), and far edges also improve, though more modestly (+0.31 labeled and +0.43 unlabeled points). By contrast, adjacent edges are slightly negative on average (-0.89 labeled and -0.29 unlabeled points). We do not read this as evidence that the model is intrinsically better at

TABLE 1 – Main results on the test set : document-level macro F1 (%, mean ± std over 3 seeds). The temporal-only columns exclude the structural anchoring label DEPEND-ON.

	All labels		Temporal-only	
Model	**Labeled**	**Unlabeled**	**Labeled**	**Unlabeled**
Baseline (T=0)	66.56 ± 0.19	73.97 ± 0.31	60.99 ± 0.16	71.53 ± 0.44
Iterative (T=3)	$\mathbf{68.95 \pm 0.80}$	$\mathbf{76.06 \pm 0.81}$	$\mathbf{63.98 \pm 0.71}$	$\mathbf{73.96 \pm 0.77}$
Delta	**+2.38**	**+2.09**	**+2.99**	**+2.43**

all long-distance decisions. Rather, it is consistent with the idea that iterative message passing is most useful when the initial local representation is not yet sufficient and the model benefits from conditioning on a broader predicted temporal context.

Per-relation trends point in the same direction. Averaged over the three iterative runs, the largest gains from i=0 to i=3 appear for AFTER (+2.51 points), OVERLAP (+1.48 points), and BEFORE (+1.37 points). DEPEND-ON changes little (+0.11 points), and INCLUDED is slightly negative on average (-0.27 points), with mixed behavior across seeds. Iterative refinement therefore does not act as a uniform improvement mechanism across all temporal relations. Instead, it appears to help most when relation prediction benefits from revising earlier edge hypotheses after contextual information has been propagated through the predicted graph.

4.3 Final Test Results

Having established the development-set refinement pattern, we now report final performance on the test set over 3 seeds. Table 1 reports document-level macro F1 on the test set (mean ± std over 3 seeds). It shows that the iterative model improves over the matched baseline on both primary metrics : +2.38 percentage points in labeled document-level macro F1 and +2.09 points in unlabeled document-level macro F1 on the test set. The gain is consistent across all three paired seeds. For labeled document-level macro F1, the iterative model improves by +2.12, +1.77, and +3.26 points over seeds 30, 31, and 32 ; for unlabeled document-level macro F1, the corresponding gains are +2.02, +1.16, and +3.09 points.

The temporal-only panel shows that the same pattern holds when DEPEND-ON is excluded. In the TDG annotation scheme, DEPEND-ON is used for structural attachments, including event attachment to ROOT when no reference event is annotated and timex anchoring to the document creation time (DCT), rather than to express temporal ordering or inclusion between content mentions (Yao *et al.*, 2020). The iterative model still improves by +2.99 labeled points and +2.43 unlabeled points on average, which indicates that the benefit is not limited to structural attachment decisions in the full all-label score.

4.4 Comparison to Prior Work

To contextualize our results, Table 2 compares our labeled TDG scores with two earlier TDG parsers. These systems differ substantially in task formulation and modeling budget, so the table is intended as contextual reference rather than as a direct leaderboard.

TABLE 2 – Contextual reference on general-TDG using reported labeled F1 scores. The NLI-based row reports the average over 5 seeds from Yao *et al.* (2023) ; the DocTime row is the reported score from Mathur *et al.* (2022) ; our row reports 3-seed labeled document-level macro F1.

Model	**Dev.**	**Test**
NLI-based TDG (avg.)	66.0	74.0
DocTime	69.0	77.0
Ours	59.24	68.95

The comparison is not direct : the NLI-based parser uses a different formulation, casting TDG parsing as parent selection by verbalizing possible child–parent relations as NLI hypotheses ; for the TDG data, it fine-tunes RoBERTa-large-MNLI, restricts candidate parents with a window covering more than 99% of cases, and appends textual descriptions of the distance between the child and candidate parent to the NLI input. DocTime is graph-based, but it includes several additional modeling components : separate graph encoders over structural, syntactic, and semantic graphs, Graph U-Net, and auxiliary path-prediction supervision. Our system is intentionally narrower : it is a controlled graph-based parser designed to isolate the effect of iterative refinement under a matched training and evaluation setup. For that reason, the main empirical claim of this paper remains the comparison between the matched iterative and single-pass models, not the absolute gap to these prior systems.

5 Conclusion

This paper asked whether iterative refinement improves document-level temporal dependency parsing compared to a single-pass baseline. Our iterative model (T=3) improves over the matched baseline (T=0) by +2.38 points in labeled document-level macro F1 and +2.09 points in unlabeled document-level macro F1 on the test set. The per-iteration analyses on the development set further show monotonic gains and increasing self-Jaccard stability, which is consistent with a refinement process in which early iterations make broader corrections and later iterations consolidate an increasingly stable graph. Taken together, these results support iterative refinement as a useful modeling choice for document-level temporal dependency parsing.

This study also has clear scope limitations. We evaluate on a single TDG dataset and focus on a controlled comparison between single-pass and iterative parsing, leaving broader validation across datasets, encoders, and architectural variants for future work. Our goal has been to isolate the effect of iterative refinement in a controlled graph-based TDG parser, rather than to maximize absolute benchmark performance with additional resources, auxiliary objectives, or more elaborate task formulations. In addition, decoding uses a development-calibrated threshold rather than a constrained graph-level decoder, so structural properties such as acyclicity or transitivity are not explicitly enforced.

Références

BETHARD S., KOLOMIYETS O. & MOENS M.-F. (2012). Annotating story timelines as temporal

dependency structures. In N. CALZOLARI, K. CHOUKRI, T. DECLERCK, M. U. DOĞAN, B. MAEGAARD, J. MARIANI, A. MORENO, J. ODIJK & S. PIPERIDIS, Éds., *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, p. 2721–2726, Istanbul, Turkey : European Language Resources Association (ELRA).

CASSIDY T., MCDOWELL B., CHAMBERS N. & BETHARD S. (2014). An annotation framework for dense event ordering. In K. TOUTANOVA & H. WU, Éds., *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, p. 501–506, Baltimore, Maryland : Association for Computational Linguistics. DOI : 10.3115/v1/P14-2082.

CHAMBERS N. & JURAFSKY D. (2008). Jointly combining implicit constraints improves temporal ordering. In M. LAPATA & H. T. NG, Éds., *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, p. 698–706, Honolulu, Hawaii : Association for Computational Linguistics.

CHATURVEDI R., BAGHERSHAHI P., MEDYA S. & DI EUGENIO B. (2025). Temporal relation extraction in clinical texts : A span-based graph transformer approach. In W. CHE, J. NABENDE, E. SHUTOVA & M. T. PILEHVAR, Éds., *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 25765–25788, Vienna, Austria : Association for Computational Linguistics. DOI : 10.18653/v1/2025.acl-long.1251.

CHEN Y., WU L. & ZAKI M. (2020). Iterative deep graph learning for graph neural networks : Better and robust node embeddings. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, p. 19314–19326.

DENIS P. & MULLER P. (2011). Predicting globally-coherent temporal structures from texts via endpoint inference and graph decomposition. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, p. 1788–1793.

DEVLIN J., CHANG M.-W., LEE K. & TOUTANOVA K. (2019). BERT : Pre-training of deep bidirectional transformers for language understanding. In J. BURSTEIN, C. DORAN & T. SOLORIO, Éds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, p. 4171–4186, Minneapolis, Minnesota : Association for Computational Linguistics. DOI : 10.18653/v1/N19-1423.

DOZAT T. & MANNING C. D. (2018). Simpler but more accurate semantic dependency parsing. In I. GUREVYCH & Y. MIYAO, Éds., *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, p. 484–490, Melbourne, Australia : Association for Computational Linguistics. DOI : 10.18653/v1/P18-2077.

FLOQUET N., ROUX J. L., TOMEH N. & CHARNOIS T. (2025). Scaling graph-based dependency parsing with arc vectorization and attention-based refinement. In L. CHIRUZZO, A. RITTER & L. WANG, Éds., *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics : Human Language Technologies (Volume 2 : Short Papers)*, p. 722–734, Albuquerque, New Mexico : Association for Computational Linguistics. DOI : 10.18653/v1/2025.naacl-short.60.

HUANG Q., HU Y., ZHU S., FENG Y., LIU C. & ZHAO D. (2023). More than classification : A unified framework for event temporal relation extraction. In A. ROGERS, J. BOYD-GRABER & N. OKAZAKI, Éds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 9631–9646, Toronto, Canada : Association for Computational Linguistics. DOI : 10.18653/v1/2023.acl-long.536.

JANG E., GU S. & POOLE B. (2017). Categorical reparameterization with gumbel-softmax.

LIN T.-Y., GOYAL P., GIRSHICK R., HE K. & DOLLÁR P. (2020). Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **42**(2), 318–327. DOI : 10.1109/TPAMI.2018.2858826.

MATHUR P., MORARIU V., KAYNIG-FITTKAU V., GU J., DERNONCOURT F., TRAN Q., NENKOVA A., MANOCHA D. & JAIN R. (2022). DocTime : A document-level temporal dependency graph parser. In M. CARPUAT, M.-C. DE MARNEFFE & I. V. MEZA RUIZ, Éds., *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, p. 993–1009, Seattle, United States : Association for Computational Linguistics. DOI : 10.18653/v1/2022.naacl-main.73.

MOHAMMADSHAHI A. & HENDERSON J. (2021). Recursive non-autoregressive graph-to-graph transformer for dependency parsing with iterative refinement. *Transactions of the Association for Computational Linguistics*, **9**, 120–138. DOI : 10.1162/tacl_a_00358.

PUSTEJOVSKY J., CASTAÑO J. M., INGRIA R., SAURÍ R., GAIZAUSKAS R. J., SETZER A., KATZ G. & RADEV D. R. (2004). Timeml : Robust specification of event and temporal expressions in text. In M. T. MAYBURY, Éd., *New Directions in Question Answering*, p. 28–34. Cambridge, MA : MIT Press.

ROSS H., CAI J. & MIN B. (2020). Exploring Contextualized Neural Language Models for Temporal Dependency Parsing. In B. WEBBER, T. COHN, Y. HE & Y. LIU, Éds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 8548–8553, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2020.emnlp-main.689.

RUSH A. & PETROV S. (2012). Vine pruning for efficient multi-pass dependency parsing. In E. FOSLER-LUSSIER, E. RILOFF & S. BANGALORE, Éds., *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, p. 498–507, Montréal, Canada : Association for Computational Linguistics.

YAO J., BETHARD S., WRIGHT-BETTNER K., GOLDNER E., HARRIS D. & SAVOVA G. (2023). Textual entailment for temporal dependency graph parsing. In T. NAUMANN, A. BEN ABACHA, S. BETHARD, K. ROBERTS & A. RUMSHISKY, Éds., *Proceedings of the 5th Clinical Natural Language Processing Workshop*, p. 191–199, Toronto, Canada : Association for Computational Linguistics. DOI : 10.18653/v1/2023.clinicalnlp-1.25.

YAO J., QIU H., MIN B. & XUE N. (2020). Annotating Temporal Dependency Graphs via Crowdsourcing. In B. WEBBER, T. COHN, Y. HE & Y. LIU, Éds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 5368–5380, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2020.emnlp-main.432.

ZHANG Y. & XUE N. (2018). Structured interpretation of temporal relations. In N. CALZOLARI, K. CHOUKRI, C. CIERI, T. DECLERCK, S. GOGGI, K. HASIDA, H. ISAHARA, B. MAEGAARD, J. MARIANI, H. MAZO, A. MORENO, J. ODIJK, S. PIPERIDIS & T. TOKUNAGA, Éds., *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan : European Language Resources Association (ELRA).