Extracting Narrative Timelines as Temporal Dependency Structures

Oleksandr Kolomiyets
KU Leuven
Celestijnenlaan 200A
B-3001 Heverlee, Belgium
Oleksandr.Kolomiyets@
cs.kuleuven.be

Steven Bethard
University of Colorado
Campus Box 594
Boulder, CO 80309, USA
Steven.Bethard@
colorado.edu

Marie-Francine Moens
KU Leuven
Celestijnenlaan 200A
B-3001 Heverlee, Belgium
Sien.Moens@
cs.kuleuven.be

Abstract

We propose a new approach to characterizing the timeline of a text: temporal dependency structures, where all the events of a narrative are linked via partial ordering relations like BEFORE, AFTER, OVERLAP and IDENTITY. We annotate a corpus of children's stories with temporal dependency trees, achieving agreement (Krippendorff's Alpha) of 0.856 on the event words, 0.822 on the links between events, and of 0.700 on the ordering relation labels. We compare two parsing models for temporal dependency structures, and show that a deterministic non-projective dependency parser outperforms a graph-based maximum spanning tree parser, achieving labeled attachment accuracy of 0.647 and labeled tree edit distance of 0.596. Our analysis of the dependency parser errors gives some insights into future research directions.

1 Introduction

There has been much recent interest in identifying events, times and their relations along the timeline, from event and time ordering problems in the TempEval shared tasks (Verhagen et al., 2007; Verhagen et al., 2010), to identifying time arguments of event structures in the Automated Content Extraction program (Linguistic Data Consortium, 2005; Gupta and Ji, 2009), to timestamping event intervals in the Knowledge Base Population shared task (Artiles et al., 2011; Amigó et al., 2011).

However, to date, this research has produced fragmented document timelines, because only specific types of temporal relations in specific contexts have been targeted. For example, the TempEval tasks only looked at relations between events in the same or adjacent sentences (Verhagen et al., 2007; Verhagen et al., 2010), and the Automated Content Extraction program only looked at time arguments for specific types of events, like *being born* or *transferring money*.

In this article, we propose an approach to temporal information extraction that identifies a single connected timeline for a text. The temporal language in a text often fails to specify a total ordering over all the events, so we annotate the timelines as temporal dependency structures, where each event is a node in the dependency tree, and each edge between nodes represents a temporal ordering relation such as BEFORE, AFTER, OVERLAP or IDENTITY. We construct an evaluation corpus by annotating such temporal dependency trees over a set of children's stories. We then demonstrate how to train a timeline extraction system based on dependency parsing techniques instead of the pair-wise classification approaches typical of prior work.

The main contributions of this article are:

- We propose a new approach to characterizing temporal structure via dependency trees.
- We produce an annotated corpus of temporal dependency trees in children's stories.
- We design a non-projective dependency parser for inferring timelines from text.

The following sections first review some relevant prior work, then describe the corpus annotation and the dependency parsing algorithm, and finally present our evaluation results.

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 88–97,
Jeju, Republic of Korea, 8-14 July 2012.

2 Related Work

Much prior work on the annotation of temporal information has constructed corpora with incomplete timelines. The TimeBank (Pustejovsky et al., 2003b; Pustejovsky et al., 2003a) provided a corpus annotated for all events and times, but temporal relations were only annotated when the relation was judged to be salient by the annotator. In the TempEval competitions (Verhagen et al., 2007; Verhagen et al., 2010), annotated texts were provided for a few different event and time configurations, for example, an event and a time in the same sentence, or two main-clause events from adjacent sentences. Bethard et al. (2007) proposed to annotate temporal relations one syntactic construction at a time, producing an initial corpus of only verbal events linked to events in subordinated clauses. One notable exception to this pattern of incomplete timelines is the work of Bramsen et al. (2006) where temporal structures were annotated as directed acyclic graphs. However they worked on a much coarser granularity, annotating not the ordering between individual events, but between multi-sentence segments of text.

In part because of the structure of the available training corpora, most existing temporal information extraction models formulate temporal linking as a pair-wise classification task, where each pair of events and/or times is examined and classified as having a temporal relation or not. Early work on the TimeBank took this approach (Boguraev and Ando, 2005), classifying relations between all events and times within 64 tokens of each other. Most of the top-performing systems in the TempEval competitions also took this pair-wise classification approach for both event-time and event-event temporal relations (Bethard and Martin, 2007; Cheng et al., 2007; UzZaman and Allen, 2010; Llorens et al., 2010). Systems have also tried to take advantage of more global information to ensure that the pair-wise classifications satisfy temporal logic transitivity constraints, using frameworks such as integer linear programming and Markov logic networks (Bramsen et al., 2006; Chambers and Jurafsky, 2008; Yoshikawa et al., 2009; UzZaman and Allen, 2010). Yet the basic approach is still centered around pair-wise classifications, not the complete temporal structure of a document.

Our work builds upon this prior research, both improving the annotation approach to generate the fully connected timeline of a story, and improving the models for timeline extraction using dependency parsing techniques. We use the annotation scheme introduced in more detail in Bethard et. al. (2012), which proposes to annotate temporal relations as dependency links between head events and dependent events. This annotation scheme addresses the issues of incoherent and incomplete annotations by guaranteeing that all events in a plot are connected along a single timeline. These connected timelines allow us to design new models for timeline extraction in which we jointly infer the temporal structure of the text and the labeled temporal relations. We employ methods from syntactic dependency parsing, adapting them to our task by including features typical of temporal relation labeling models.

3 Corpus Annotation

The corpus of stories for children was drawn from the fables collection of (McIntyre and Lapata, 2009)[1] and annotated as described in (Bethard et al., 2012). In this section we illustrate the main annotation principles for coherent temporal annotation. As an example story, consider:

> Two Travellers were on the road together, when a Bear suddenly appeared on the scene. Before he observed them, one made for a tree at the side of the road, and climbed up into the branches and hid there. The other was not so nimble as his companion; and, as he could not escape, he threw himself on the ground and pretended to be dead... [37.txt]

Figure 1 shows the temporal dependency structure that we expect our annotators to identify in this story.

The annotators were provided with guidelines both for which kinds of words should be identified as events, and for which kinds of events should be linked by temporal relations. For identifying event words, the standard TimeML guidelines for annotating events (Pustejovsky et al., 2003a) were augmented with two additional guidelines:

[1]Data available at http://homepages.inf.ed.ac.uk/s0233364/McIntyreLapata09/

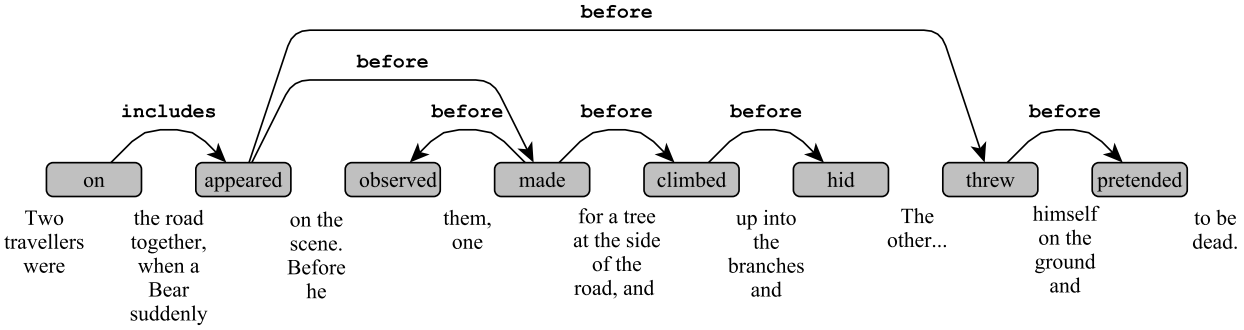

Figure 1: Event timeline for the story of the Travellers and the Bear. Nodes are events and edges are temporal relations. Edges denote temporal relations signaled by linguistic cues in the text. Temporal relations that can be inferred via transitivity are not shown.

- Skip negated, modal or hypothetical events (e.g. *could not escape*, *dead* in *pretended to be dead*).
- For phrasal events, select the single word that best paraphrases the meaning (e.g. in *used to snap* the event should be *snap*, in *kept perfectly still* the event should be *still*).

For identifying the temporal dependencies (i.e. the ordering relations between event words), the annotators were instructed to link each event in the story to a single nearby event, similar to what has been observed in reading comprehension studies (Johnson-Laird, 1980; Brewer and Lichtenstein, 1982). When there were several reasonable nearby events to choose from, the annotators were instructed to choose the temporal relation that was easiest to infer from the text (e.g. preferring relations with explicit cue words like *before*). A set of six temporal relations was used: BEFORE, AFTER, INCLUDES, IS-INCLUDED, IDENTITY or OVERLAP.

Two annotators annotated temporal dependency structures in the first 100 fables of the McIntyre-Lapata collection and measured inter-annotator agreement by Krippendorff's Alpha for nominal data (Krippendorff, 2004; Hayes and Krippendorff, 2007). For the resulting annotated corpus annotators achieved Alpha of 0.856 on the event words, 0.822 on the links between events, and of 0.700 on the ordering relation labels. Thus, we concluded that the temporal dependency annotation paradigm was reliable, and the resulting corpus of 100 fables[2] could be used to train a temporal dependency parsing model.

[2]Available from `http://www.bethard.info/data/fables-100-temporal-dependency.xml`

4 Parsing Models

We consider two different approaches to learning a temporal dependency parser: a shift-reduce model (Nivre, 2008) and a graph-based model (McDonald et al., 2005). Both models take as input a sequence of event words and produce as output a tree structure where the events are linked via temporal relations. Formally, a parsing model is a function $(W \rightarrow \Pi)$ where $W = w_1 w_2 \dots w_n$ is a sequence of event words, and $\pi \in \Pi$ is a dependency tree $\pi = (V, E)$ where:

- $V = W \cup \{Root\}$, that is, the vertex set of the graph is the set of words in W plus an artificial root node.
- $E = \{(w_h, r, w_d) : w_h \in V, w_d \in V, r \in R = \{\text{BEFORE, AFTER, INCLUDES, IS\_INCLUDED, IDENTITY, OVERLAP}\}\}$, that is, in the edge set of the graph, each edge is a link between a dependent word and its head word, labeled with a temporal relation.
- $(w_h, r, w_d) \in E \implies w_d \neq Root$, that is, the artificial root node has no head.
- $(w_h, r, w_d) \in E \implies ((w'_h, r', w_d) \in E \implies w_h = w'_h \wedge r = r')$, that is, for every node there is at most one head and one relation label.
- E contains no (non-empty) subset of arcs $(w_h, r_i, w_i), (w_i, r_j, w_j), \dots, (w_k, r_l, w_h)$, that is, there are no cycles in the graph.

SHIFT	Move all of L_2 and the head of Q onto L_1
	$([a_1 \dots a_i], [b_1 \dots b_j], [w_k w_{k+1} \dots], E) \rightarrow ([a_1 \dots a_i b_1 \dots b_j w_k], [], [w_{k+1} \dots], E)$
NO-ARC	Move the head of L_1 to the head of L_2
	$([a_1 \dots a_i a_{i+1}], [b_1 \dots b_j], Q, E) \rightarrow ([a_1 \dots a_i], [a_{i+1} b_1 \dots b_j], Q, E)$
LEFT-ARC	Create a relation where the head of L_1 depends on the head of Q
	Not applicable if a_{i+1} is the root or already has a head, or if there is a path connecting w_k and a_{i+1}
	$([a_1 \dots a_i a_{i+1}], [b_1 \dots b_j], [w_k \dots], E) \rightarrow ([a_1 \dots a_i], [a_{i+1} b_1 \dots b_j], [w_k \dots], E \cup (w_k, r, a_{i+1})$
RIGHT-ARC	Create a relation where the head of Q depends on the head of L_1
	Not applicable if w_k is the root or already has a head, or if there is a path connecting w_k and a_{i+1}
	$([a_1 \dots a_i a_{i+1}], [b_1 \dots b_j], [w_k \dots], E) \rightarrow ([a_1 \dots a_i], [a_{i+1} b_1 \dots b_j], [w_k \dots], E \cup (a_{i+1}, r, w_k)$

Table 1: Transition system for Covington-style shift-reduce dependency parsers.

4.1 Shift-Reduce Parsing Model

Shift-reduce dependency parsers start with an input queue of unlinked words, and link them into a tree by repeatedly choosing and performing actions like shifting a node to a stack, or popping two nodes from the stack and linking them. Shift-reduce parsers are typically defined in terms of configurations and a transition system, where the configurations describe the current internal state of the parser, and the transition system describes how to get from one state to another. Formally, a deterministic shift-reduce dependency parser is defined as $(C, T, C_F, \text{INIT}, \text{TREE})$ where:

- C is the set of possible parser configurations c_i
- $T \subseteq (C \rightarrow C)$ is the set of transitions t_i from one configuration c_j to another c_{j+1} allowed by the parser
- $\text{INIT} \in (W \rightarrow C)$ is a function from the input words to an initial parser configuration
- $C_F \subseteq C$ are the set of final parser configurations c_F where the parser is allowed to terminate
- $\text{TREE} \in (C_F \rightarrow \Pi)$ is a function that extracts a dependency tree π from a final parser state c_F

Given this formalism and an oracle $o \in (C \rightarrow T)$, which can choose a transition given the current configuration of the parser, dependency parsing can be accomplished by Algorithm 1. For temporal dependency parsing, we adopt the Covington set of transitions (Covington, 2001) as it allows for parsing the non-projective trees, which may also contain "crossing" edges, that occasionally occur in our annotated corpus. Our parser is therefore defined as:

Algorithm 1 Deterministic parsing with an oracle.

```
c ← INIT(W)
while c ∉ C_F do
    t ← o(c)
    c ← t(c)
end while
return TREE(c)
```

- $c = (L_1, L_2, Q, E)$ is a parser configuration, where L_1 and L_2 are lists for temporary storage, Q is the queue of input words, and E is the set of identified edges of the dependency tree.
- $T = \{\text{SHIFT}, \text{NO-ARC}, \text{LEFT-ARC}, \text{RIGHT-ARC}\}$ is the set of transitions described in Table 1.
- $\text{INIT}(W) = ([Root], [], [w_1, w_2, \dots, w_n], \emptyset)$ puts all input words on the queue and the artificial root on L_1.
- $C_F = \{(L_1, L_2, Q, E) \in C : L_1 = \{W \cup \{Root\}\}, L_2 = Q = \emptyset\}$ accepts final states where the input words have been moved off of the queue and lists and into the edges in E.
- $\text{TREE}((L_1, L_2, Q, E)) = (W \cup \{Root\}, E)$ extracts the final dependency tree.

The oracle o is typically defined as a machine learning classifier, which characterizes a parser configuration c in terms of a set of features. For temporal dependency parsing, we learn a Support Vector Machine classifier (Yamada and Matsumoto, 2003) using the features described in Section 5.

4.2 Graph-Based Parsing Model

One shortcoming of the shift-reduce dependency parsing approach is that each transition decision

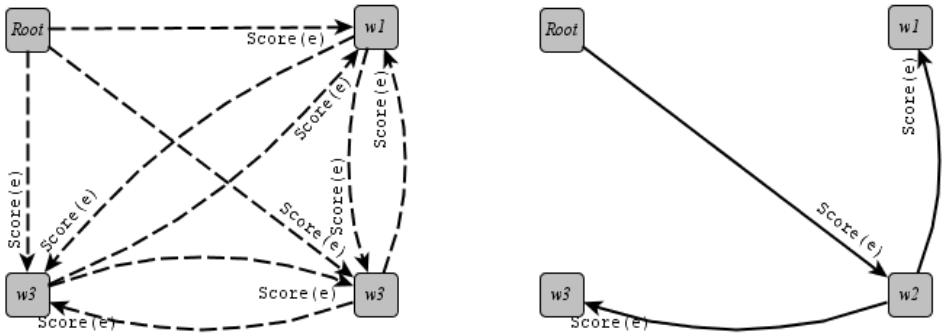

Figure 2: A setting for the graph-based parsing model: an initial dense graph G (left) with edge scores $\text{SCORE}(e)$. The resulting dependency tree as a spanning tree with the highest score over the edges (right).

made by the model is final, and cannot be revisited to search for more globally optimal trees. Graph-based models are an alternative dependency parsing model, which assembles a graph with weighted edges between all pairs of words, and selects the tree-shaped subset of this graph that gives the highest total score (Fig. 2). Formally, a graph-based parser follows Algorithm 2, where:

- $W' = W \cup \{Root\}$
- $\text{SCORE} \in ((W' \times R \times W) \rightarrow \Re)$ is a function for scoring edges
- SPANNINGTREE is a function for selecting a subset of edges that is a tree that spans over all the nodes of the graph.

Algorithm 2 Graph-based dependency parsing

$E \leftarrow \{(e, \text{SCORE}(e)) : e \in (W' \times R \times W))\}$
$G \leftarrow (W', E)$
return $\text{SPANNINGTREE}(G)$

The SPANNINGTREE function is usually defined using one of the efficient search techniques for finding a maximum spanning tree. For temporal dependency parsing, we use the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967) which solves this problem by iteratively selecting the edge with the highest weight and removing edges that would create cycles. The result is the globally optimal maximum spanning tree for the graph (Georgiadis, 2003).

The SCORE function is typically defined as a machine learning model that scores an edge based on a set of features. For temporal dependency parsing, we learn a model to predict edge scores via the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003; Crammer et al., 2006) using the set of features defined in Section 5.

5 Feature Design

The proposed parsing algorithms both rely on machine learning methods. The shift-reduce parser (SRP) trains a machine learning classifier as the oracle $o \in (C \rightarrow T)$ to predict a transition t from a parser configuration $c = (L_1, L_2, Q, E)$, using node features such as the heads of L_1, L_2 and Q, and edge features from the already predicted temporal relations in E. The graph-based maximum spanning tree (MST) parser trains a machine learning model to predict $\text{SCORE}(e)$ for an edge $e = (w_i, r_j, w_k)$, using features of the nodes w_i and w_k. The full set of features proposed for both parsing models, derived from the state-of-the-art systems for temporal relation labeling, is presented in Table 2. Note that both models share features that look at the nodes, while only the shift-reduce parser has features for previously classified edges.

6 Evaluations

Evaluations were performed using 10-fold cross-validation on the fables annotated in Section 3. The corpus contains 100 fables, a total of 14,279 tokens and a total of 1136 annotated temporal relations. As

Feature	SRP	MST
Word	$\surd^*$	$\surd^*$
Lemma	$\surd^*$	$\surd^*$
Part of speech (POS) tag	$\surd^*$	$\surd^*$
Suffixes	$\surd^*$	$\surd^*$
Syntactically governing verb	$\surd^*$	$\surd^*$
Governing verb lemma	$\surd^*$	$\surd^*$
Governing verb POS tag	$\surd^*$	$\surd^*$
Governing verb POS suffixes	$\surd^*$	$\surd^*$
Prepositional phrase occurrence	$\surd^*$	$\surd^*$
Dominated by auxiliary verb?	$\surd^*$	$\surd^*$
Dominated by modal verb?	$\surd^*$	$\surd^*$
Temporal signal word is nearby?	$\surd^*$	$\surd^*$
Head word lemma	$\surd^*$	$\surd^*$
Temporal relation labels of a_i and its leftmost and rightmost dependents	$\surd$	
Temporal relation labels of a_{i-1}'s leftmost and rightmost dependents	$\surd$	
Temporal relation labels of b_1 and its leftmost and rightmost dependents	$\surd$	

Table 2: Features for the shift-reduce parser (SRP) and the graph-based maximum spanning tree (MST) parser. The $\surd^*$ features are extracted from the heads of L_1, L_2 and Q for SRP and from each node of the edge for MST.

only 40 instances of OVERLAP relations were annotated when neither INCLUDES nor IS_INCLUDED label matched, for evaluation purposes all instances of these relations were merged into the temporally coarse OVERLAP relation. Thus, the total number of OVERLAP relations in the corpus grew from 40 to 258 annotations in total.

To evaluate the parsing models (SRP and MST) we proposed two baselines. Both are based on the assumption of linear temporal structures of narratives as the temporal ordering process that was evidenced by studies in human text rewriting (Hickmann, 2003). The proposed baselines are:

- **LinearSeq**: A model that assumes all events occur in the order they are written, adding links between each pair of adjacent events, and labeling all links with the relation BEFORE.

- **ClassifySeq**: A model that links each pair of adjacent events, but trains a pair-wise classifier to predict the relation label for each pair. The classifier is a support vector machine trained using the same features as the MST parser. This is an approximation of prior work, where the pairs of events to classify with a temporal relation were given as an input to the system. (Note that Section 6.2 will show that for our corpus, applying the model only to adjacent pairs of events is quite competitive for just getting the basic unlabeled link structure right.)

The Shift-Reduce parser (SRP; Section 4.1) and the graph-based, maximum spanning tree parser (MST; Section 4.2) are compared to these baselines.

6.1 Evaluation Criteria and Metrics

Model performance was evaluated using standard evaluation criteria for parser evaluations:

Unlabeled Attachment Score (UAS) The fraction of events whose head events were correctly predicted. This measures whether the correct pairs of events were linked, but not if they were linked by the correct relations.

Labeled Attachment Score (LAS) The fraction of events whose head events were correctly predicted with the correct relations. This measures both whether the correct pairs of events were linked and whether their temporal ordering is correct.

Tree Edit Distance In addition to the UAS and LAS the *tree edit distance* score has been recently introduced for evaluating dependency structures (Tsarfaty et al., 2011). The tree edit distance score for a tree π is based on the following operations $\lambda \in \Lambda : \Lambda = \{\text{DELETE}, \text{INSERT}, \text{RELABEL}\}$:

- $\lambda =$DELETE delete a non-root node v in π with parent u, making the children of v the children of u, inserted in the place of v as a subsequence in the left-to-right order of the children of u.

- $\lambda =$INSERT insert a node v as a child of u in π making it the parent of a consecutive subsequence of the children of u.

- $\lambda =$RELABEL change the label of node v in π

Any two trees π_1 and π_2 can be turned one into another by a sequence of edit operations $\{\lambda_1, ..., \lambda_n\}$.

	UAS	LAS	UTEDS	LTEDS
LinearSeq	0.830	0.581	0.689	0.549
ClassifySeq	0.830	0.581	0.689	0.549
MST	0.837	0.614*	0.710	0.571
SRP	0.830	0.647*†	0.712	0.596*

Table 3: Performance levels of temporal structure parsing methods. A * indicates that the model outperforms LinearSeq and ClassifiedSeq at $p < 0.01$ and a † indicates that the model outperforms MST at $p < 0.05$.

Error Type	Num.	%
OVERLAP → BEFORE	24	43.7
Attach to further head	18	32.7
Attach to nearer head	6	11.0
Other types of errors	7	12.6
Total	55	100

Table 4: Error distribution from the analysis of 55 errors of the Shift-Reduce parsing model.

Taking the shortest such sequence, the tree edit distance is calculated as the sum of the edit operation costs divided by the size of the tree (i.e. the number of words in the sentence). For temporal dependency trees, we assume each operation costs 1.0. The final score subtracts the edit distance from 1 so that a perfect tree has score 1.0. The *labeled* tree edit distance score (LTEDS) calculates sequences over the tree with all its labeled temporal relations, while the *unlabeled* tree edit distance score (UTEDS) treats all edges as if they had the same label.

6.2 Results

Table 3 shows the results of the evaluation. The unlabeled attachment score for the LinearSeq baseline was 0.830, suggesting that annotators were most often linking adjacent events. At the same time, the labeled attachment score was 0.581, indicating that even in fables, the stories are not simply linear, that is, there are many relations other than BEFORE. The ClassifySeq baseline performs identically to the LinearSeq baseline, which shows that the simple pairwise classifier was unable to learn anything beyond predicting all relations as BEFORE.

In terms of labeled attachment score, both dependency parsing models outperformed the baseline models – the maximum spanning tree parser achieved 0.614 LAS, and the shift-reduce parser achieved 0.647 LAS. The shift-reduce parser also outperformed the baseline models in terms of labeled tree edit distance, achieving 0.596 LTEDS vs. the baseline 0.549 LTEDS. These results indicate that dependency parsing models are a good fit to our whole-story timeline extraction task.

Finally, in comparing the two different dependency parsing models, we observe that the shift-reduce parser outperforms the maximum spanning tree parser in terms of labeled attachment score (0.647 vs. 0.614). It has been argued that graph-based models like the maximum spanning tree parser should be able to produce more globally consistent and correct dependency trees, yet we do not observe that here. A likely explanation for this phenomenon is that the shift-reduce parsing model allows for features describing previous parse decisions (similar to the incremental nature of human parse decisions), while the joint nature of the maximum spanning tree parser does not.

6.3 Error Analysis

To better understand the errors our model is still making, we examined two folds (55 errors in total in 20% of the evaluation data) and identified the major categories of errors:

- **OVERLAP → BEFORE**: The model predicts the correct head, but predicts its label as BEFORE, while the correct label is OVERLAP.
- **Attach to further head**: The model predicts the wrong head, and predicts as the head an event that is further away than the true head.
- **Attach to nearer head**: The model predicts the wrong head, and predicts as the head an event that is closer than the true head.

Table 4 shows the distribution of the errors over these categories. The two most common types of errors, OVERLAP → BEFORE and *Attach to further head*, account for 76.4% of all the errors.

The most common type of error is predicting a BEFORE relation when the correct answer is an OVERLAP relation. Figure 3 shows an example of such an error, where the model predicts that the Spendthrift *stood* before he *saw*, while the annotator indicates that the seeing happened during the

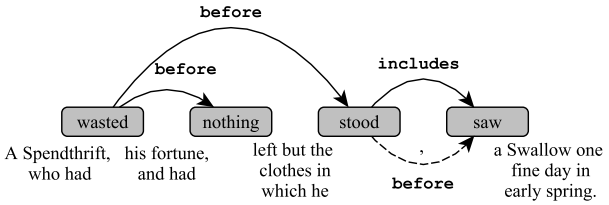

Figure 3: An OVERLAP → BEFORE parser error. True links are solid lines; the parser error is the dotted line.

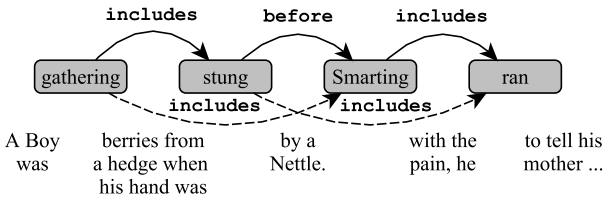

Figure 4: Parser errors attaching to further away heads. True links are solid lines; parser errors are dotted lines.

time in which he was standing. An analysis of these OVERLAP → BEFORE errors suggests that they occur in scenarios like this one, where the duration of one event is significantly longer than the duration of another, but there are no direct cues for these duration differences. We also observe these types of errors when one event has many sub-events, and therefore the duration of the main event typically includes the durations of all the sub-events. It might be possible to address these kinds of errors by incorporating automatically extracted event duration information (Pan et al., 2006; Gusev et al., 2011).

The second most common error type of the model is the prediction of a head event that is further away than the head identified by the annotators. Figure 4 gives an example of such an error, where the model predicts that the *gathering* includes the *smarting*, instead of that the *gathering* includes the *stung*. The second error in the figure is also of the same type. In 65% of the cases where this type of error occurs, it occurs after the parser had already made a label classification error such as BEFORE → OVERLAP. So these errors may be in part due to the sequential nature of shift-reduce parsing, where early errors propagate and cause later errors.

7 Discussion and Conclusions

In this article, we have presented an approach to temporal information extraction that represents the timeline of a story as a temporal dependency tree. We have constructed an evaluation corpus where such temporal dependencies have been annotated over a set of 100 children's stories. We have introduced two dependency parsing techniques for extracting story timelines and have shown that both outperform a rule-based baseline and a prior-work-inspired pair-wise classification baseline. Comparing the two dependency parsing models, we have found that a shift-reduce parser, which more closely mirrors the incremental processing of our human annotators, outperforms a graph-based maximum spanning tree parser. Our error analysis of the shift-reduce parser revealed that being able to estimate differences in event durations may play a key role in improving parse quality.

We have focused on children's stories in this study, in part because they typically have simpler temporal structures (though not so simple that our rule-based baseline could parse them accurately). In most of our fables, there were only one or two characters with at most one or two simultaneous sequences of actions. In other domains, the timeline of a text is likely to be more complex. For example, in clinical records, descriptions of patients may jump back and forth between the patient history, the current examination, and procedures that have not yet happened.

In future work, we plan to investigate how to best apply the dependency structure approach to such domains. One approach might be to first group events into their *narrative containers* (Pustejovsky and Stubbs, 2011), for example, grouping together all events linked to the time of a patient's examination. Then within each narrative container, our dependency parsing approach could be applied. Another approach might be to join the individual timeline trees into a document-wide tree via discourse relations or relations to the document creation time. Work on how humans incrementally process such timelines in text may help to decide which of these approaches holds the most promise.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive comments. This research was partially funded by the TERENCE project (EU FP7-257410) and the PARIS project (IWT SBO 110067).

References

[Amigó et al.2011] Enrique Amigó, Javier Artiles, Qi Li, and Heng Ji. 2011. An evaluation framework for aggregated temporal information extraction. In *SIGIR-2011 Workshop on Entity-Oriented Search*.

[Artiles et al.2011] Javier Artiles, Qi Li, Taylor Cassidy, Suzanne Tamang, and Heng Ji. 2011. CUNY_BLENDER TAC-KBP2011 temporal slot filling system description. In *Text Analytics Conference (TAC2011)*.

[Bethard and Martin2007] Steven Bethard and James H. Martin. 2007. CU-TMP: Temporal relation classification using syntactic and semantic features. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 129–132, Prague, Czech Republic, June. ACL.

[Bethard et al.2007] Steven Bethard, James H. Martin, and Sara Klingenstein. 2007. Finding temporal structure in text: Machine learning of syntactic temporal relations. *International Journal of Semantic Computing (IJSC)*, 1(4):441–458, 12.

[Bethard et al.2012] Steven Bethard, Oleksandr Kolomiyets, and Marie-Francine Moens. 2012. Annotating narrative timelines as temporal dependency structures. In *Proceedings of the International Conference on Linguistic Resources and Evaluation*, Istanbul, Turkey, May. ELRA.

[Boguraev and Ando2005] Branimir Boguraev and Rie Kubota Ando. 2005. TimeBank-driven TimeML analysis. In *Annotating, Extracting and Reasoning about Time and Events*. Springer.

[Bramsen et al.2006] P. Bramsen, P. Deshpande, Y.K. Lee, and R. Barzilay. 2006. Inducing temporal graphs. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 189–198. ACL.

[Brewer and Lichtenstein1982] William F. Brewer and Edward H. Lichtenstein. 1982. Stories are to entertain: A structural-affect theory of stories. *Journal of Pragmatics*, 6(5-6):473 – 486.

[Chambers and Jurafsky2008] N. Chambers and D. Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 698–706. ACL.

[Cheng et al.2007] Yuchang Cheng, Masayuki Asahara, and Yuji Matsumoto. 2007. NAIST.Japan: Temporal relation identification using dependency parsed tree. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 245–248, Prague, Czech Republic, June. ACL.

[Chu and Liu1965] Y. J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, pages 1396–1400.

[Covington2001] M.A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102.

[Crammer and Singer2003] K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

[Crammer et al.2006] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

[Edmonds1967] J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, pages 233–240.

[Georgiadis2003] L. Georgiadis. 2003. Arborescence optimization problems solvable by Edmonds' algorithm. *Theoretical Computer Science*, 301(1-3):427–437.

[Gupta and Ji2009] Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ACLShort '09, pages 369–372, Stroudsburg, PA, USA. ACL.

[Gusev et al.2011] Andrey Gusev, Nathanael Chambers, Divye Raj Khilnani, Pranav Khaitan, Steven Bethard, and Dan Jurafsky. 2011. Using query patterns to learn the duration of events. In *Proceedings of the International Conference on Computational Semantics*, pages 145–154.

[Hayes and Krippendorff2007] A.F. Hayes and K. Krippendorff. 2007. Answering the call for a standard reliability measure for coding data. *Communication Methods and Measures*, 1(1):77–89.

[Hickmann2003] Maya Hickmann. 2003. *Children's Discourse: Person, Space and Time Across Languages*. Cambridge University Press, Cambridge, UK.

[Johnson-Laird1980] P.N. Johnson-Laird. 1980. Mental models in cognitive science. *Cognitive Science*, 4(1):71–115.

[Krippendorff2004] K. Krippendorff. 2004. *Content analysis: An introduction to its methodology*. Sage Publications, Inc.

[Linguistic Data Consortium2005] Linguistic Data Consortium. 2005. ACE (Automatic Content Extraction) English annotation guidelines for events version 5.4.3 2005.07.01.

[Llorens et al.2010] Hector Llorens, Estela Saquete, and Borja Navarro. 2010. TIPSem (English and Spanish): Evaluating CRFs and semantic roles in TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 284–291, Uppsala, Sweden, July. ACL.

[McDonald et al.2005] R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. ACL.

[McIntyre and Lapata2009] N. McIntyre and M. Lapata. 2009. Learning to tell tales: A data-driven approach to story generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 217–225. ACL.

[Nivre2008] J. Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

[Pan et al.2006] Feng Pan, Rutu Mulkar, and Jerry R. Hobbs. 2006. Learning event durations from event descriptions. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 393–400, Sydney, Australia, July. ACL.

[Pustejovsky and Stubbs2011] J. Pustejovsky and A. Stubbs. 2011. Increasing informativeness in temporal annotation. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 152–160. ACL.

[Pustejovsky et al.2003a] James Pustejovsky, José Castaño, Robert Ingria, Roser Saurý, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003a. TimeML: Robust specification of event and temporal expressions in text. In *Proceedings of the Fifth International Workshop on Computational Semantics (IWCS-5)*, Tilburg.

[Pustejovsky et al.2003b] James Pustejovsky, Patrick Hanks, Roser Saurý, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. 2003b. The TimeBank corpus. In *Proceedings of Corpus Linguistics*, pages 647–656.

[Tsarfaty et al.2011] R. Tsarfaty, J. Nivre, and E. Andersson. 2011. Evaluating dependency parsing: Robust and heuristics-free cross-annotation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 385–396. ACL.

[UzZaman and Allen2010] Naushad UzZaman and James Allen. 2010. TRIPS and TRIOS system for TempEval-2: Extracting temporal information from text. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 276–283, Uppsala, Sweden, July. ACL.

[Verhagen et al.2007] Marc Verhagen, Robert Gaizauskas, Frank Schilder, Graham Katz, and James Pustejovsky. 2007. SemEval2007 Task 15: TempEval temporal relation identification. In *SemEval-2007: 4th International Workshop on Semantic Evaluations*.

[Verhagen et al.2010] Marc Verhagen, Roser Saurí, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 57–62, Stroudsburg, PA, USA. ACL.

[Yamada and Matsumoto2003] H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*.

[Yoshikawa et al.2009] K. Yoshikawa, S. Riedel, M. Asahara, and Y. Matsumoto. 2009. Jointly identifying temporal relations with Markov Logic. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 405–413. ACL.