Deriving Timelines from Text

Mathias Landhäußer, Tobias Hey, and Walter F. Tichy
June 3, 2014
Background

- Almost everyone has one or more programmable devices. There are literally billions of chips out there.

- But only a tiny fraction of the owners of programmable devices can program.

- Programmability, the most fundamental aspect of computers, is inaccessible to almost everyone.
The AliceNLP Project

Objectives

- Synthesize programs from natural language.
- Targeted system: CMU’s Alice

Target audience: People who do not “speak” a programming language.
- Programming level: Scripting, but no programming of complex algorithms.
The AliceNLP Project
Approach

- Learn what Alice can do (→ do not hard-code system functionality)
- Read a textual description
- And map it to Alice’s functionality

Diagram:
- Alice API
- English Script
- Ontology
- WordNet Cyc etc.
- Animation
- Alice Program

Mathias Landhäußer, Tobias Hey, and Walter F. Tichy | Deriving Time Lines from Text
The AliceNLP Project
The Corpus – An Empirical Project Driver

Empirical Study (a.k.a. Building the Alice Corpus)
- Start with the animation!
- Let people describe animations in their own words.
- Now we have scripts and the program we want to translate them into.

The corpus drives AliceNLP
- Analyze users’ language.
- Identify challenges.
- Create a benchmark to test the system.

Identified challenges so far
- Scene setup
- Parallelism
- Level of abstraction
- Reordering of actions
Solutions from AI and NLP

- Programming in natural language
  - User-centered programming [Pane&Myers]
  - NLC [Ballard&Biermann], Natural Java [Price], Metafor [Liu&Lieberman]
  - Pegasus [Knöll&Menzini]
  - Robotics

- NL Understanding
  - Detect absolute points of time (dates) and references (e.g. yesterday)
  - Build question answering systems; e.g. [Pustejovsky]
  - Put documents (or events thereof) on the global time line; e.g. [Schilder], identify time spans (“noon ‘till midnight” → 12 hours) [Ohlbach]

- Temporal Reasoning
  - Event calculus [Russel]
Non-Sequentiality

- Deviations from sequential order use **temporal expressions**: Tense, temporal adverbs, and temporal prepositions

- Tense alone is not a useful indicator of order.

- Temporal adverbs and temporal prepositions encode order:
  - Before
  - At the beginning
  - After
  - Etc.

→ Identify temporal patterns with signal words adverbs and prepositions
Patterns for Temporal Expressions

- Translate each NL pattern into 
  \( \text{operator(} \text{anchor action, transfer action)} \)
- 3 operators
  - \( \text{after}(a, b) \)
  - \( \text{before}(a, b) \)
  - \( \text{at}(n, a) \)

Examples
- Do e and then do f. \( \rightarrow \text{after}(e, f) \)
- At the end, do g. \( \rightarrow \text{before}(\omega, g) \)
Workflow of our Analysis

Before Mathias asks for questions, he gives a presentation and then…

- Search for signal words → “Before”
- Search for actions
  → a: “…asks …”
  → b: “…gives …”
- Temporal Pattern → before a, do b
- Operator → before(a, b)

NLP Processing → Extract Temporal Expressions → Detect Sub-Clauses → Determine Anchor and Transfer Actions → Apply Operator
Evaluation

- 3 different animations
  - Animation “Bunny”: Control texts without rearrangements
  - Other texts make heavy use of rearrangements

<table>
<thead>
<tr>
<th>Animation</th>
<th>Texts</th>
<th>TEs</th>
<th>✓</th>
<th>✗</th>
<th>!</th>
<th>↔</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bunny</td>
<td>4</td>
<td>16</td>
<td>15</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Cheerleader</td>
<td>10</td>
<td>81</td>
<td>67</td>
<td>5</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>Dragon</td>
<td>10</td>
<td>69</td>
<td>60</td>
<td>1</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td>24</td>
<td>166</td>
<td>142</td>
<td>7</td>
<td>17</td>
<td>9</td>
</tr>
</tbody>
</table>

- Source of errors (✗)
  - 3/7: Parser error
  - 1/7: Subject placed one action before *and* after another action

- Source of misses (!): 13/17 stem from one author only

✓ correct, ✗ misinterpreted, ! missed

↔ # rearrangements needed to correct time line
Examples for Failures

- Missed conjunctions
  - „The Bunny jumps upward three times and then bends forward, lies down on the meadow and eats the mushroom.“
  - Parser gives conjunction between jumps, bends, and eats only.

- Missed references between actions
  - After the cheerleader speaks to the penguin, it turns its head right. After turning its head, the penguin flaps its wings once.
  - The Bunny hops twice. ... Before the Bunny hops twice the Frog croaks and then jumps away.
Conclusion & Future Work

- Determining the correct order of actions is essential for programming.
- Simple – yet effective – heuristics helps in reordering actions.

Future work

- Co-referencing actions
- More temporal expressions / patterns
- Parallel actions
- Control structures
References


### Temporal Expressions and Anchor Actions

<table>
<thead>
<tr>
<th>Temporal Expression</th>
<th>Anchor Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>before (at the beginning of the phrase)</td>
<td>1st action of the phrase</td>
</tr>
<tr>
<td>before (in the middle of a phrase)</td>
<td>Directly following action</td>
</tr>
<tr>
<td>before (with already mentioned action)</td>
<td>Mentioned action</td>
</tr>
<tr>
<td>before (entity + synonym of <em>started</em>)</td>
<td>Previous action of this entity (before the temporal expression)</td>
</tr>
<tr>
<td>before (that / this)</td>
<td>Last action of previous (sub-)phrase</td>
</tr>
<tr>
<td>but first / previously</td>
<td>Last action of previous (sub-)phrase</td>
</tr>
<tr>
<td>after (at the beginning of the phrase)</td>
<td>1st action of the phrase</td>
</tr>
<tr>
<td>after (in the middle of the phrase)</td>
<td>Directly following action</td>
</tr>
<tr>
<td>after (with already mentioned action)</td>
<td>Mentioned action</td>
</tr>
<tr>
<td>after (entity + synonym of <em>finished</em>)</td>
<td>Previous action of this entity (before the temporal expression)</td>
</tr>
<tr>
<td>after (that / this)</td>
<td>Last action of previous (sub-)phrase</td>
</tr>
</tbody>
</table>
# Temporal Expressions and Anchor Actions

<table>
<thead>
<tr>
<th>Temporal Expression</th>
<th>Anchor Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>at the end / finally</td>
<td>ω</td>
</tr>
<tr>
<td>at the beginning / start</td>
<td>α</td>
</tr>
<tr>
<td>Afterwards, then, later (on), there- / where- / hereupon, thereafter, followed by</td>
<td>Previous action</td>
</tr>
<tr>
<td>By the time / when (entity + synonym of finished)</td>
<td>Previous action of this entity (before the temporal expression)</td>
</tr>
<tr>
<td>As (first, second, …)</td>
<td>Position # on time line</td>
</tr>
<tr>
<td>As (his/her/its first, second, …)</td>
<td>Position # on time line</td>
</tr>
</tbody>
</table>