Karlsruher Institut für Technologie

Deriving Timelines from Text

Mathias Landhäußer, Tobias Hey, and Walter F. Tichy

June 3, 2014

RAISE'14

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

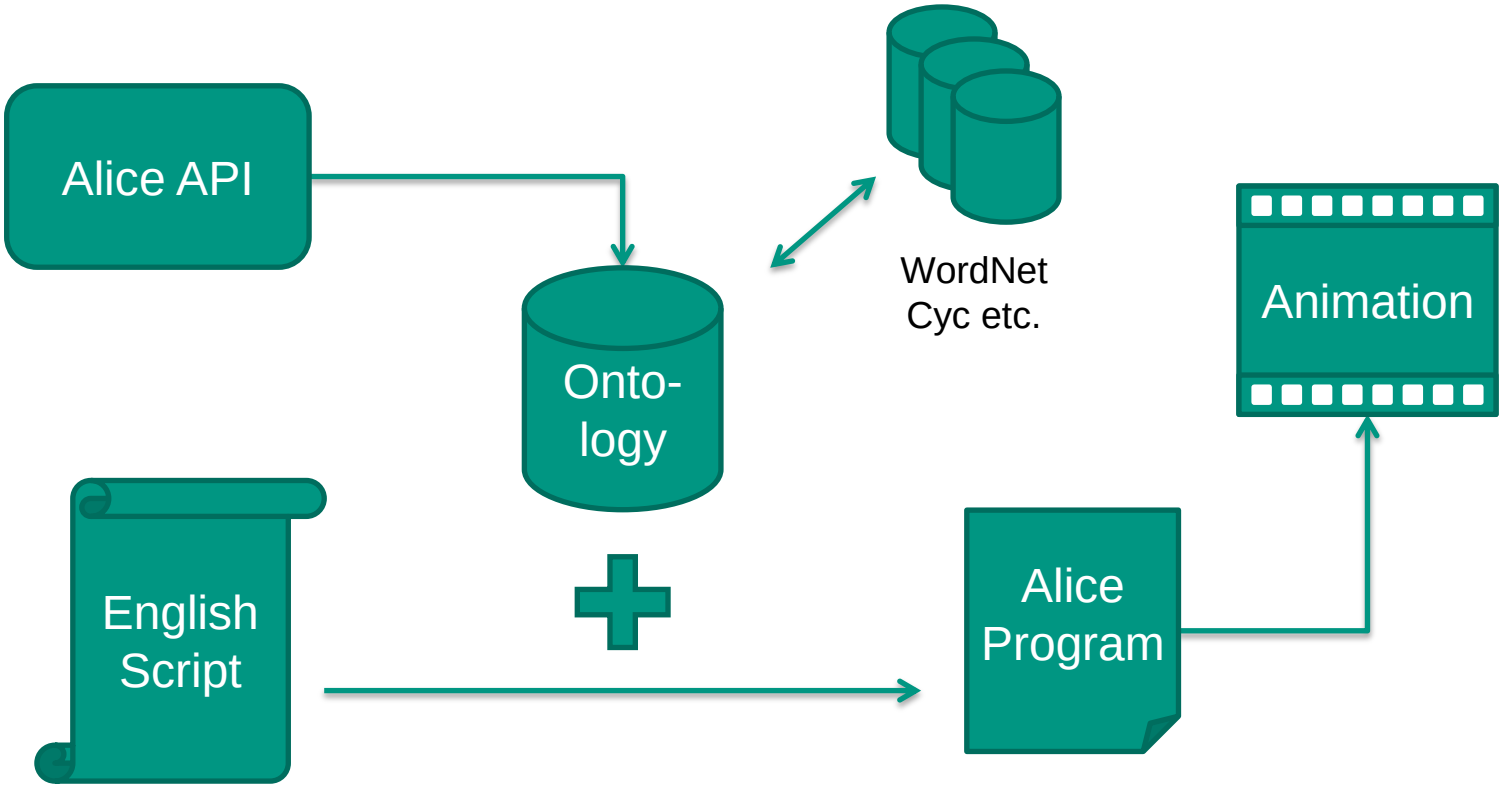
- Synthesize programs from natural language.
- Targeted system: CMU's Alice

- Target audience: People who do not "speak" a programming language.
- Programming level: Scripting, but no programming of complex algorithms.

[Conway]

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



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

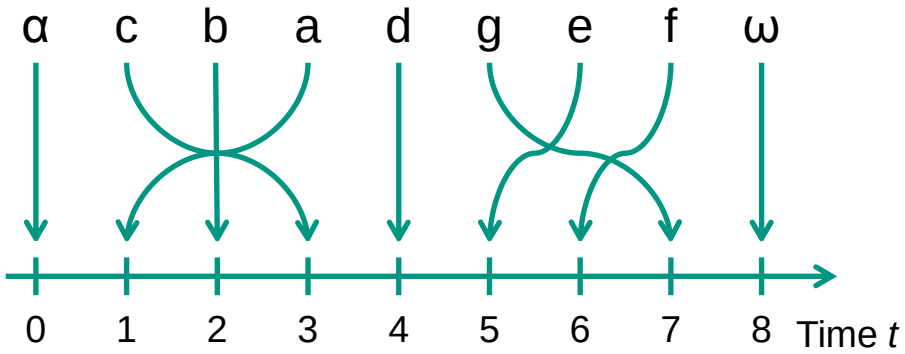
- Tense alone is not a useful indicator of order.

- 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 *operator(anchor action, transfer action)*
- 3 operators
  - *after(a, b)*
  - *before(a, b)*
  - *at(n, a)*

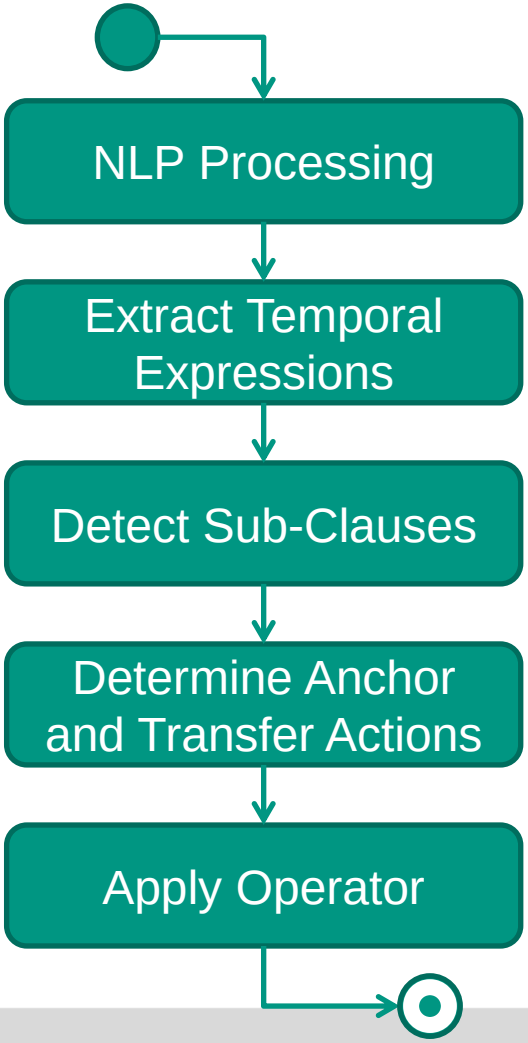


- Examples
  - Do e and then do f. → *after(e, f)*
  - At the end, do g. → *before(ω, g)*

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)*



Evaluation

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

Animation	Texts	TEs	✓	✗	!	↔
Bunny	4	16	15	1	0	1
Cheerleader	10	81	67	5	9	5
Dragon	10	69	60	1	8	2
Total	24	166	142 (86%)	7 (4%)	17 (10%)	9

- 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

- [Ballard&Biermann] B. W. Ballard and A. W. Biermann. Programming in natural language: NLC as a prototype. In Proceedings of the 1979 annual conference, ACM '79, pages 228–237, New York, NY, USA, 1979. ACM.
- [Conway] M. J. Conway. Alice: Easy-to-Learn 3D Scripting for Novices. PhD thesis, Faculty of the School of Engineering and Applied Science, University of Virginia, Dec. 1997.
- [Liu&Lieberman] H. Liu and H. Lieberman. Metafor: visualizing stories as code. In Proceedings of the 10th int. conference on Intelligent user interfaces, pages 305–307, New York, NY, USA, 2005. ACM.
- [Ohlbach] H. J. Ohlbach. Computational treatment of temporal notions: The cttn-system. In F. Schilder, G. Katz, and J. Pustejovsky, editors, Annotating, Extracting and Reasoning about Time and Events, volume 4795 of Lecture Notes in Computer Science, pages 72–87. Springer Berlin Heidelberg, 2007.
- [Pane&Myers] J. F. Pane, B. A. Myers, and C. A. Ratanamahatana. Studying the language and structure in non-programmers' solutions to programming problems. Int. J. Hum.-Comput. Stud., 54(2):237–264, Feb. 2001.
- [Price] D. Price, E. Rilofff, J. Zachary, and B. Harvey. NaturalJava: a natural language interface for programming in Java. In Proceedings of the 5th international conference on Intelligent user interfaces, IUI '00, pages 207–211, New York, NY, USA, 2000. ACM.
- [Pustejovsky] J. Pustejovsky et al. The specification language TimeML. The language of time: A reader, pages 545–557, 2005.
- [Russel] S. Russell and P. Norvig. The Artificial Intelligence. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2010.
- [Schilder] F. Schilder. Event extraction and temporal reasoning in legal documents. In F. Schilder, G. Katz, and J. Pustejovsky, editors, Annotating, Extracting and Reasoning about Time and Events, volume 4795 of Lecture Notes in Computer Science, pages 59–71. Springer Berlin Heidelberg, 2007.

Temporal Expressions and Anchor Actions

Temporal Expression	Anchor Action
before (at the beginning of the phrase)	1st action of the phrase
before (in the middle of a phrase)	Directly following action
before (with already mentioned action)	Mentioned action
before (entity + synonym of *started*)	Previous action of this entity (before the temporal expression)
before (that / this)	Last action of previous (sub-)phrase
but first / previously	Last action of previous (sub-)phrase
after (at the beginning of the phrase)	1st action of the phrase
after (in the middle pf the phrase)	Directly following action
after (with already mentioned action)	Mentioned action
after (entity + synonym of *finished*)	Previous action of this entity (before the temporal expression)
after (that / this)	Last action of previous (sub-)phrase

Temporal Expressions and Anchor Actions

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