

jTLEX: a Java Library for Timeline Extraction(ANAnSI)

Student: Anthony Nguyen

Mentor: Antonela Radas, Project Manager / Mark Finlayson, Product Owner

Instructor: Dr. Masoud Sadjadi, Florida International University

Problem

Qualitative temporal graphs that are derived from annotations on text| e.g., TimeMLannotations| explicitly reveal only partial orderings of events and times.

For many purposes, however, a total ordering (i.e., a timeline) is more useful. We adapt prior work on solving point algebra problems to the task of extracting multiple timelines from TimeML annotated texts and demonstrate, for the first time, an exact, end-to-end solution, which we call TLEX (TimeLine EXtraction).

In collaboration with MITRE, we must convert the ANAnSI, also known as Advanced Narrative Analytics System Infrastructure, prototype system output to the TimeML standards.

Contributions

Before mapping from ANAnSI to TimeML each member was tasked with developing a class for a certain part of language and a parser pertaining to that class. I oversaw:

- Clause class
- Clause class Junit test
- Clause parser
- Clause parser test

I had also collaborated with:

- Adaptor to ANAnSI system

Current System

The purpose of the ANAnSI is to map the ANAnSI data structure to TimeML structure for Timeline extraction.

The current system is the backend code which parses through a json file and extracts the information needed to create the ANAnSI data structure, once the data structure is created with the information extracted, the system maps the ANAnSI data structure to TimeML data structure.

Once the mapping is done, the system will do the timeline extraction and prints some output.

Screenshots

```

MATCHED LINES
ANAnSI lid: 1775 firstNode: say secondNode: become relType: EVIDENTIAL
TimeML lid: 29 firstNode: say secondNode: become relType: EVIDENTIAL
ANAnSI lid: 2802 firstNode: learned secondNode: taken relType: EVIDENTIAL
TimeML lid: 28 firstNode: learned secondNode: taken relType: FACTIVE

LEFTOVER LINES FROM ANAnSI
ANAnSI lid: 2112 firstNode: was secondNode: flew relType: AFTER
ANAnSI lid: 1283 firstNode: say secondNode: fight relType: EVIDENTIAL
ANAnSI lid: 2112 firstNode: was secondNode: included relType: EVIDENTIAL
ANAnSI lid: 2837 firstNode: chosen secondNode: were relType: SIMULTANEOUS
ANAnSI lid: 1286 firstNode: is secondNode: say relType: EVIDENTIAL
ANAnSI lid: 2854 firstNode: was secondNode: were relType: SIMULTANEOUS
ANAnSI lid: 2831 firstNode: say secondNode: held relType: EVIDENTIAL

LEFTOVER LINES FROM TIMEML
TimeML lid: 14 firstNode: picked secondNode: 19980304 relType: BEFORE
TimeML lid: 11 firstNode: held secondNode: now relType: ENDED_BY
TimeML lid: 26 firstNode: included secondNode: twenty relType: END
TimeML lid: 17 firstNode: learned secondNode: today relType: IS_INCLUDED
TimeML lid: 2 firstNode: mission secondNode: December relType: IS_INCLUDED
TimeML lid: 24 firstNode: named secondNode: 19980304 relType: AFTER
TimeML lid: 6 firstNode: included secondNode: chosen relType: AFTER
TimeML lid: 4 firstNode: along secondNode: models relType: BEFORE
TimeML lid: 9 firstNode: commander secondNode: boss relType: IDENTITY
TimeML lid: 8 firstNode: commander secondNode: elison relType: SIMULTANEOUS
TimeML lid: 32 firstNode: That secondNode: flew relType: AFTER
TimeML lid: 12 firstNode: lack secondNode: held relType: DURING
TimeML lid: 13 firstNode: picked secondNode: astronaut relType: BEGINS
TimeML lid: 31 firstNode: is secondNode: now relType: BEFORE
TimeML lid: 21 firstNode: become secondNode: starting relType: BEFORE
TimeML lid: 38 firstNode: become secondNode: part relType: FACTIVE
TimeML lid: 4 firstNode: his time secondNode: mission relType: IDENTITY
TimeML lid: 7 firstNode: along secondNode: mass relType: IDENTITY
TimeML lid: 25 firstNode: chosen secondNode: twenty relType: BEGINS
TimeML lid: 3 firstNode: co-pilot secondNode: 19980304 relType: BEFORE
TimeML lid: 19 firstNode: progression secondNode: held relType: INCLUDES
TimeML lid: 18 firstNode: progression secondNode: hurried relType: INCLUDES
TimeML lid: 16 firstNode: followed secondNode: progression relType: IDENTITY
TimeML lid: 15 firstNode: followed secondNode: picked relType: AFTER
TimeML lid: 23 firstNode: flew secondNode: flew relType: INCLUDES
TimeML lid: 27 firstNode: named secondNode: commander relType: FACTIVE
TimeML lid: 22 firstNode: 19980304 secondNode: flew relType: BEFORE
TimeML lid: 28 firstNode: say secondNode: 19980304 relType: INCLUDES

MATCHED INSTANCES
ANAnSI lid: 34 event: 33 tense: PRESENT aspect: VERB polarity: POS modality: null signal: null Event Stem: say
TimeML lid: 22 event: 25 tense: PRESENT aspect: VERB polarity: POS modality: null signal: null Event Stem: say
  
```

Figure 1: Sample output, link mapping shown above

Property on LinguisticEvent Node	Map to TimeML
EventType: "Report"	class="REPORTING"
EventIntent: "Achievement"	class="OCCURRENCE"
EventIntent: "State"	class="STATE"
EventIntent: "NegativeState"	class="STATE"
EventIntent: "PositiveAction"	class="ACTION"
EventIntent: "NegativeAction"	class="ACTION"
No EventType: "Report" and no EventIntent (default)	class="OCCURRENCE"

Figure 2: Mapping for the event classes is shown in the table above, "examples" blurred for confidential reasons

Requirements

Develop a parser that translates AnAnsi graph output to TimeML format.

- The code will accept Json files from the user and extract the information needed to create ANAnSI data structure to then be mapped into TimeML structure.
- This means it must map out all types of labels and sort them out in order to be extracted.
- Once the TimeML structure is created it will now be ready for timeline extraction.

Implementation



- Eclipse was the universal IDE all group members used to implement their code.
- Java was used to develop the backend of the program.
- JUnit framework was used to test the implementation regarding the java language.
- Subversion was used to upload the code and store it for other team members to be kept updated.
- Trello was used to set up an Agile environment where team members would be assigned task.

Resources

- 1) [Essential Scrum: A Practical Guide to the Most Popular Agile Process](#) (Chapter 2 and 4)
- 2) J. F. Allen (1983) Maintaining Knowledge about Temporal Intervals
- 3) Saurí et. al. (2006) TimeML Annotation Guidelines
- 4) Pustejovsky et. al. (2003) The TimeBank Corpus
- 5) TLEX: A Formally Correct Method for Extracting Exact Timelines from TimeML Temporal Graphs
- 6) ANANSI-TIMEML mapping schema

Summary

TLEX (TimeLine EXtraction), a new technique for extracting timelines from natural language texts that have been annotated with the TimeML markup language.

TLEX draws ideas from both NLP and the AI sub-field of temporal constraint problem solving, and we have developed, formally proved, and experimentally validated the basic components of the technique. To instantiate TLEX in a polished Java library that can be easily used by other researchers in the field, our team must map the ANAnSI data structure, a task given by MITRE, to TimeML standards in order to proceed to timeline extraction.

Takeaways

What I learned:

- How to work with a SVN within an IDE
- Create meaningful Junit 5 tests to assess implemented code
- Extensive/Advanced java language programming
- Create a parser that reads and processes information
- Slight knowledge on graph data structure Neo4j
- What it is like to work with a team on an ample project
- Learned about Timeline Extraction and its applications into real-world implementations.

Acknowledgement

The material presented in this poster is based upon the work supported by Dr. Mark Finlayson, Mustafa Ocal, and Dr. Karine Megerdooian. I'd like to thank Antonela Radas and the ANAnSI team for assistance and cooperation that I received throughout this process.