

jTLEX: a Java Library for Timeline Extraction(ANAnSI)

Student: Pablo Maldonado

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., TimeML annotations| 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 part of the data structure, then we worked on the parser of ANAnSI. I oversaw:

- Rel class
- Rel class Junit test
- Rel parser
- Rel parser test

I had also collaborated with:

- Adaptor to ANAnSI, worked on mapping from Rel (ANAnSI) to Links (TML)

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 LINKS
ANAnSI 116: 1770 firstNode: say secondNode: become relType: EVIDENTIAL
TimeML 116: 29 firstNode: say secondNode: become relType: EVIDENTIAL
ANAnSI 116: 2862 firstNode: learned secondNode: taken relType: EVIDENTIAL
TimeML 116: 28 firstNode: learned secondNode: taken relType: FACTIVE

LEFTOVER LINKS FROM ANAnSI
ANAnSI 116: 2112 firstNode: was secondNode: flew relType: AFTER
ANAnSI 116: 1322 firstNode: say secondNode: fight relType: EVIDENTIAL
ANAnSI 116: 2113 firstNode: was secondNode: included relType: EVIDENTIAL
ANAnSI 116: 2827 firstNode: chosen secondNode: were relType: SIMULTANEOUS
ANAnSI 116: 1386 firstNode: is secondNode: say relType: EVIDENTIAL
ANAnSI 116: 2854 firstNode: was secondNode: were relType: SIMULTANEOUS
ANAnSI 116: 2633 firstNode: say secondNode: held relType: EVIDENTIAL

LEFTOVER LINKS FROM TIMEML
TimeML 116: 14 firstNode: picked secondNode: 19980304 relType: BEFORE
TimeML 116: 11 firstNode: held secondNode: new relType: BEFORE
TimeML 116: 26 firstNode: included secondNode: twenty relType: ENDS
TimeML 116: 17 firstNode: learned secondNode: today relType: IS_INCLUDED
TimeML 116: 2 firstNode: mission secondNode: December relType: IS_INCLUDED
TimeML 116: 24 firstNode: named secondNode: 19980304 relType: AFTER
TimeML 116: 6 firstNode: included secondNode: chosen relType: AFTER
TimeML 116: 8 firstNode: doing secondNode: models relType: BEFORE
TimeML 116: 5 firstNode: commander secondNode: boss relType: IDENTITY
TimeML 116: 1 firstNode: commander secondNode: mission relType: SIMULTANEOUS
TimeML 116: 32 firstNode: that secondNode: flew relType: AFTER
TimeML 116: 12 firstNode: back secondNode: held relType: DURING
TimeML 116: 13 firstNode: picked secondNode: astronaut relType: BEGINS
TimeML 116: 31 firstNode: in secondNode: new relType: BEFORE
TimeML 116: 21 firstNode: become secondNode: catching relType: BEFORE
TimeML 116: 30 firstNode: become secondNode: part relType: FACTIVE
TimeML 116: 4 firstNode: this time secondNode: mission relType: IDENTITY
TimeML 116: 7 firstNode: doing secondNode: makes relType: IDENTITY
TimeML 116: 29 firstNode: chosen secondNode: twenty relType: BEGINS
TimeML 116: 3 firstNode: co-pilot secondNode: 19980304 relType: BEFORE
TimeML 116: 19 firstNode: progression secondNode: held relType: INCLUDES
TimeML 116: 18 firstNode: progression secondNode: hurried relType: INCLUDES
TimeML 116: 10 firstNode: followed secondNode: progression relType: IDENTITY
TimeML 116: 15 firstNode: followed secondNode: picked relType: AFTER
TimeML 116: 23 firstNode: flew secondNode: flew relType: INCLUDES
TimeML 116: 27 firstNode: named secondNode: commander relType: FACTIVE
TimeML 116: 22 firstNode: 19980304 secondNode: flew relType: BEFORE
TimeML 116: 20 firstNode: say secondNode: 19980304 relType: INCLUDES

MATCHED INSTANCES
ANAnSI 116: 34 event: 33 tense: PRESENT aspect: VERB polarity: POS modality: null signal: null event Stem: say
TimeML 116: 20 tense: PRESENT aspect: VERB polarity: POS modality: null signal: null event Stem: say

```

Figure 1: Sample output, link mapping shown above

```

public void convertRel2Tml(Rel rel) {
    IClause clause1 = null;
    IClause clause2 = null;
    Instance instance1 = null;
    Instance instance2 = null;
    ITimeLine timeLine = null;
    ILink link = null;

    // Identify relations between clauses
    if (rel.getType() == Type.CLAUSERELATION) {
        if (rel.getEnhancedPlusPlusDependency() == EnhancedPlusPlusDependency.CCOMP) {
            for (IClass clause = aText.getClause1()) {
                if (clause.getId() == rel.getStartNodeId()) {
                    clause1 = clause;
                } else if (clause.getId() == rel.getEndNodeId()) {
                    clause2 = clause;
                }
            }
            if (clause1 != null && clause2 != null) {
                break;
            }
        }
        if (clause1 != null && clause2 != null) {
            for (ILinguisticEvent linguisticEvent : aText.getLinguisticEvents()) {
                if (linguisticEvent.getBegin() == clause1.getPredicatePosition()) {
                    instance1 = (Instance) map.get(linguisticEvent).get(0);
                }
                else if (linguisticEvent.getBegin() == clause2.getPredicatePosition()) {
                    instance2 = (Instance) map.get(linguisticEvent).get(0);
                }
                if (instance1 != null && instance2 != null) {
                    link = new Link(linkTag.SIMV, rel.getId(), null, instance1, null, instance2, SLINKType.EVIDENTIAL);
                    links.add(link);
                    break;
                }
            }
        }
        else if (rel.getType() == Type.CLAUSERELATION) {
            if (rel.getEnhancedPlusPlusDependency() == EnhancedPlusPlusDependency.ADVCL_IP) {
                for (IClass clause = aText.getClause1()) {
                    if (clause.getId() == rel.getStartNodeId()) {
                        clause1 = clause;
                    } else if (clause.getId() == rel.getEndNodeId()) {
                        clause2 = clause;
                    }
                }
                if (clause1 != null && clause2 != null) {
                    break;
                }
            }
            if (clause1 != null && clause2 != null) {
                for (ILinguisticEvent linguisticEvent : aText.getLinguisticEvents()) {
                    if (linguisticEvent.getBegin() == clause1.getPredicatePosition()) {
                        instance1 = (Instance) map.get(linguisticEvent).get(0);
                    }
                    else if (linguisticEvent.getBegin() == clause2.getPredicatePosition()) {
                        instance2 = (Instance) map.get(linguisticEvent).get(0);
                    }
                    if (instance1 != null && instance2 != null) {
                        break;
                    }
                }
            }
        }
    }
}

```

Figure 2: Part of the code that I wrote for the Adaptor from Rel (ANAnSI) to Links (TML)

Requirements

- Develop an adaptor that translates from ANAnSI to TML:
- First, create the data structure to hold the ANAnSI structure JSON files..
 - Then, create a parser to filled that data structure given a JSON file.
 - Finally, map JSON file to TML file with adaptor.
 - 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.
- 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.

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 Eclipse
- How to create a Data Structure base on documents and mapping
- Create meaningful Junit 5 tests to assess implemented code
- 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.

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

Acknowledgement

The material presented in this poster is based upon the work supported by Dr. Mark Finlayson, Mustafa Ocal, and Dr. Karine Megerdooimian. I'd like to thank Antonela Radas for all the help that she provided to us. Finally, I would like to like to thank my project team members: Anthony Nguyen, Cristian Caro, Joshua Jimenez, Joshua Posada, and Daniel Chang for their incredible work during this semester.