Using UIMA to Structure an Open Platform for Textual Entailment

Tae-Gil Noh, Sebastian Padó
Dept. of Computational Linguistics
Heidelberg University

The paper is about

- About EXCITEMENT Open Platform
  - a suite for *Textual Entailment*
  - and, how UIMA helped us to build the platform.

- Contents of this session
  - Brief introduction to Textual Entailment, and the EXCITEMENT open platform.
  - UIMA adoption on EXCITEMENT platform.
  - Some open issues.

Textual Entailment (TE)

- A relation between two text fragments.
- Definition
  - *A text (T) entails Hypothesis (H), if a typical human reading of T would infer that H is most likely true.*
- Example
  - T: One of them is 1908 Tunguska event in Siberia, known as the Tunguska meteorite fall.
  - H1: A shooting star fell in Russia in 1908.
  - H2: Tunguska fell to Siberia in 1908.
- Typical human reading of T would say;
  - H1 is true, while H2 is not.

Textual Entailment (TE); relation on Text (T) and Hypothesis (H)

- TE is a directed relation.
- An example (directed T -> H)
  - T: John bought a Volkswagen Golf.
  - H: Now, John has a car.
    - “Textual Inference”.
- Similar to paraphrase?
  - T: He got a letter of acceptance.
  - H: The acceptance letter has been given to him.
  - Paraphrase can be regarded as a case of bidirectional entailment. (T -> H & H -> T)
- Recognizing Textual Entailment (RTE)
  - A decision task on a (Text, Hypothesis) pair.
    - ENTAILMENT or NON-ENTAILMENT

Textual Entailment (TE), as Semantic Processing Engine

- Potential of Textual Entailment (TE)
  - Various NLP applications need semantic processing.
  - But semantic processings are mostly done by application-dependent manners.
    - (vs. standardized syntactic processings)
  - TE has the potential to offer a uniform, theory-independent semantic processing.
  - Existing TE engines have been used to build proof-of-concept systems
    - Question answering, Machine Translation evaluation, Information visualization, Automatic summarization, etc.

Textual Entailment Engines

- Many different strategies
  - Tested and developed along RTE workshops.
  - The community produced several good open source systems.
- Practical problem of *Fragmentation*
  - No interoperability
    - Modules and resources are often only designed for a specific system and a specific paradigm.
  - Build-from-scratch
    - When researchers want to build a new approach, they often need to build from scratch.
    - Many of the components already exist, but not in a usable form!

Common platform for Textual Entailment?

- EXCITEMENT open platform
  - A *suite* of textual inference components.
  - Goal
    - Provide a playground of “pluggable” (reusable) TE components for the community.
    - Be the common development platform for TE researchers.
      - Like MOSES platform in Machine Translation.
  - Challenges
    - TE systems typically depends on various linguistic analysis, as well as large knowledge bases.
    - Direct source of the problem of reusability.

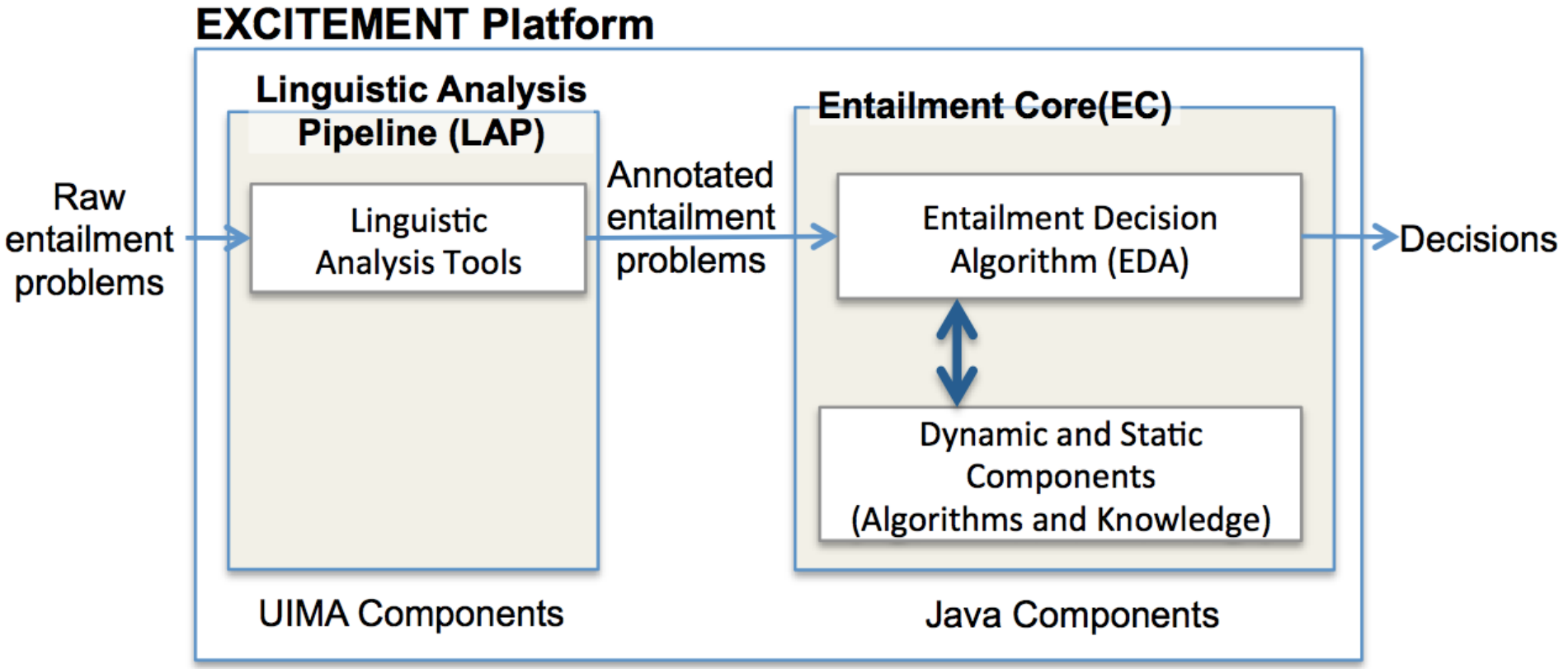
The open platform is a part of EXCITEMENT project

- EXCITEMENT - EU FP7 project
  - Home page: http://excitement-project.eu/
  - Academic and industrial partners.
- Academic side
  - Bar Ilan university (Tel aviv, BIUTEE system)
  - DFKI (Saarbrücken, TIE system)
  - FBK (Trento, EDITS system)
  - Heidelberg University
- Industrial side
  - NICE (in Israel), OMQ (in Germany), ALMA (in Italy)
  - Use the resulting TE engines of the platform for customer interaction analysis.
- First version of the platform is just out.

EXCITEMENT Open Platform

- This paper deals UIMA-related architectural aspects of EOP.
- The requirements of the platform
  - 1) Reusing of existing software
    - Easy integration of existing TE system, components and resources.
  - 2) Multilinguality
    - Adding a new language should be easy.
  - 3) Component Reusable
    - Each component is self-contained and not tied to a specific approach.
    - Should be easily replaceable, and reusable.

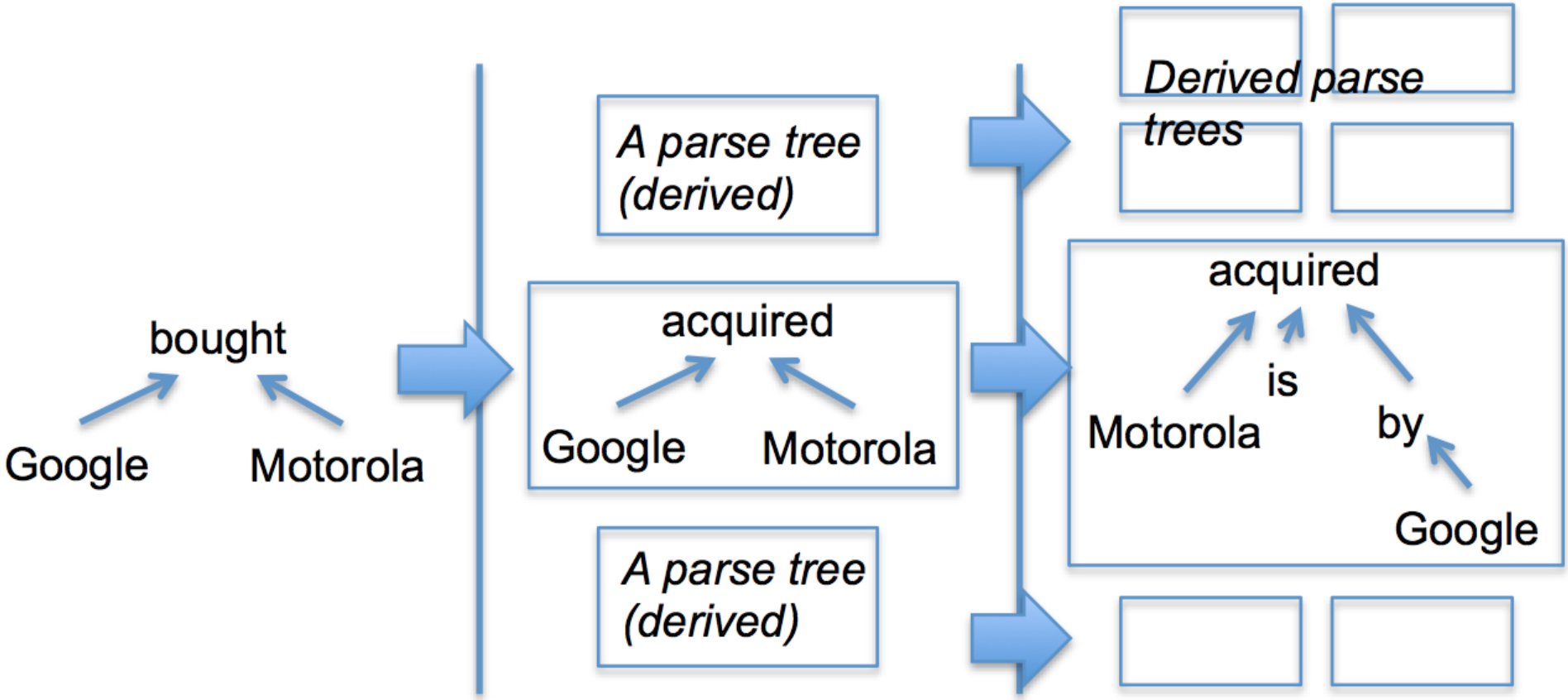
EXCITEMENT Platform Architecture Overview



EXCITEMENT Open Platform (EOP) Architecture

- UIMA adoption on EOP
  - *Partial*, and *Parallel*
- Partial
  - ***UIMA only adopted for the first part of EOP***
  - Two groups of common components in EOP
    - LAP (Linguistic analysis pipeline) & CORE
    - Only LAP part adopts UIMA
  - LAP components are naturally mapped to UIMA.
    - All component behaviors as “adding annotations”
  - Many CORE components are not natural to be treated as annotators.

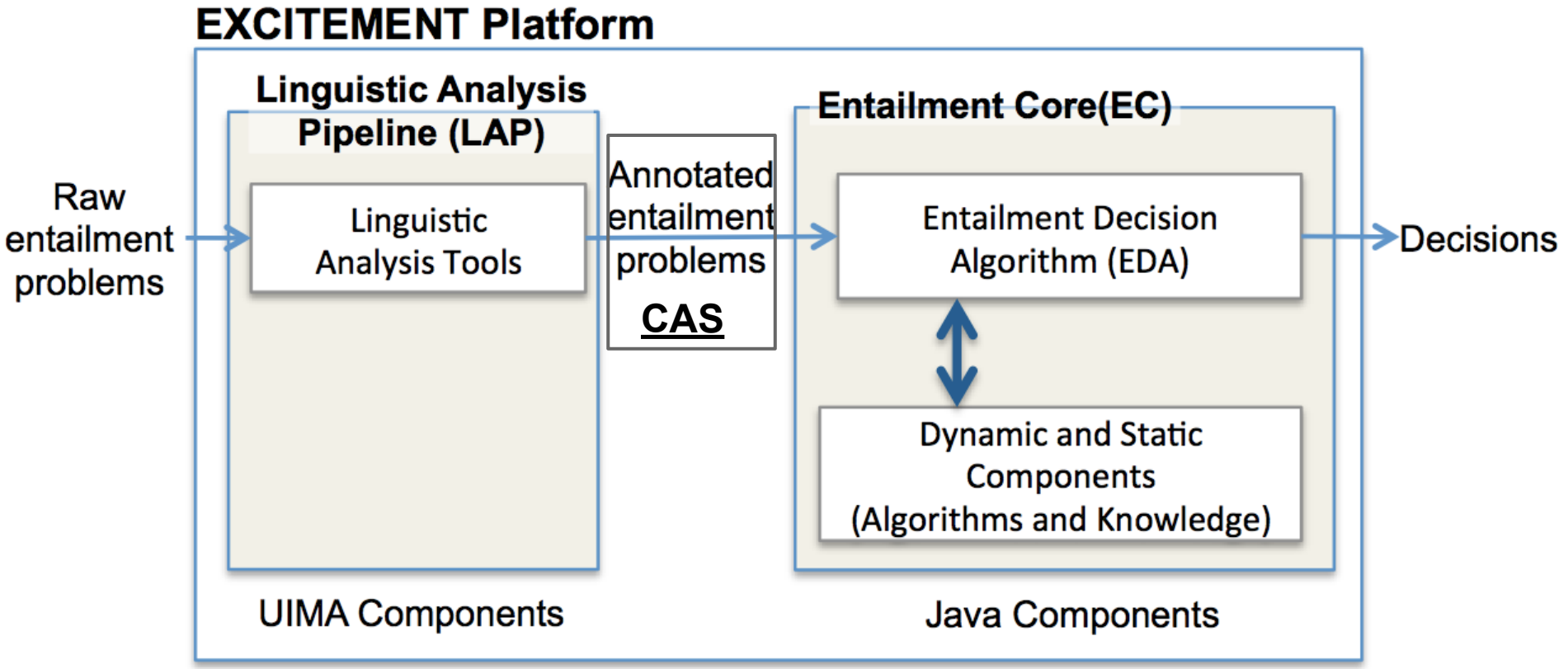
An example of Core component behavior



Core Components

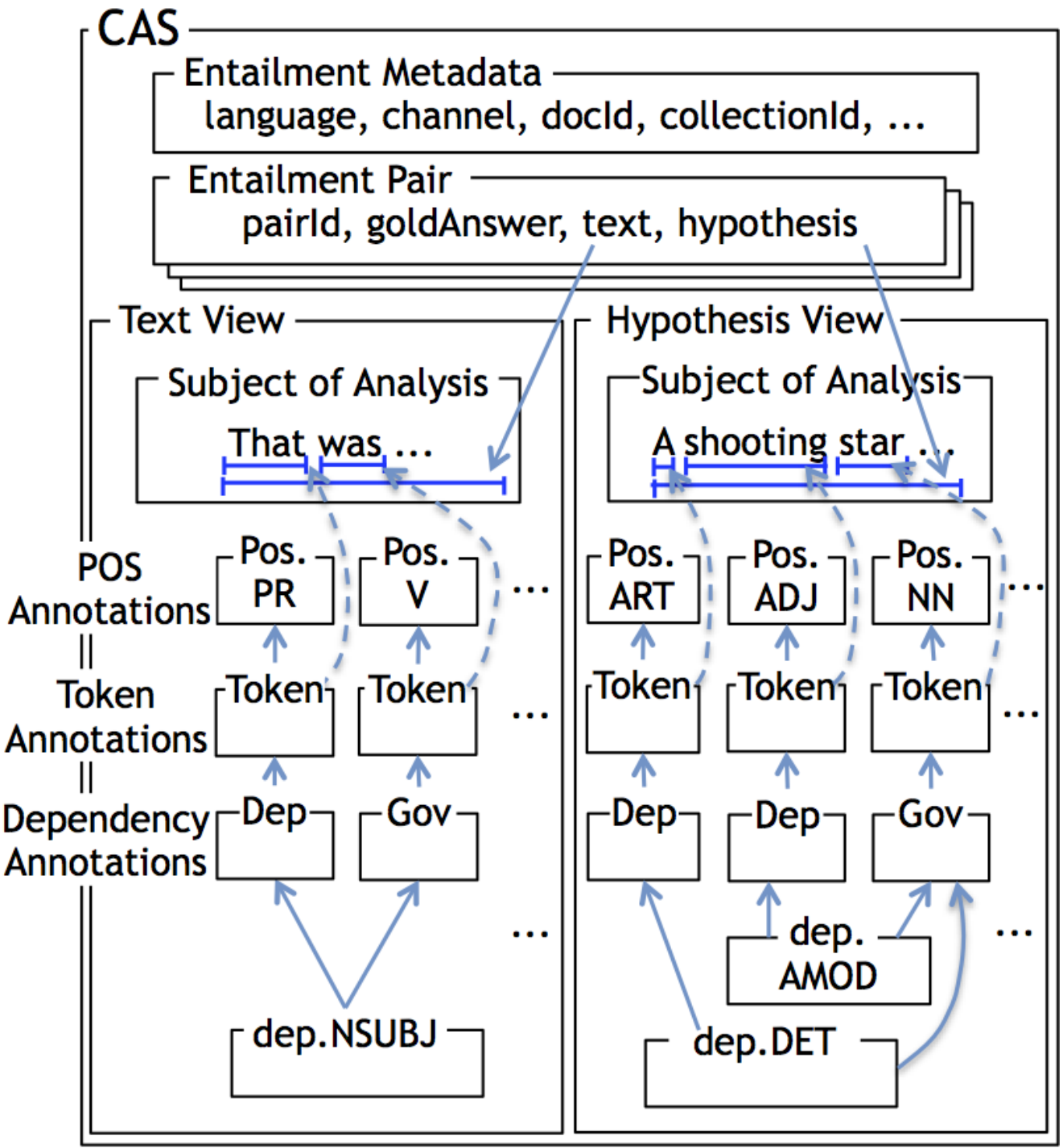
- They are defined as Java component
  - Behaviors are defined by a set of Java Interfaces, and with specific conventions.
  - However, they still use CAS (JCas) as the main data type that holds annotated data.
- Resource “look-up” components
  - Lexical Resources.
  - Syntactic-level Resources.
- Scoring components (CAS in, score out)
  - Feature Extracting components.
  - (Semantic) Distance calculation components.
- Entailment Decision components
  - EDA (Entailment Decision Algorithm).

EXCITEMENT Platform Architecture Overview



UIMA Usage in EXCITEMENT: CAS

- CAS is the central data type that connects LAP & CORE
  - CAS is “Input” to Entailment Core, and “output”of Linguistic Analysis Pipelines (LAPs).
  - Things to consider for CAS that holds TE problems
    - CAS holds a pair (t and h fragments), instead of a document.
    - Multiple text, or multiple hypothesis cases
    - Some annotations connects parts of text and hypothesis (e.g. alignment annotations)
- Two tasks on CAS adoption
  - 1) A design for T-H pair representation in CAS.
  - 2) Type systems to represent them


CAS
Entailment Metadata
language, channel, docId, collectionId, ...
Entailment Pair
pairId, goldAnswer, text, hypothesis
Text View
Subject of Analysis
That was ...
Hypothesis View
Subject of Analysis
A shooting star ...
POS Annotations
Token Annotations
Dependency Annotations
Pos. PR
Pos. V
Pos. ART
Pos. ADJ
Pos. NN
Token
Dep
Gov
dep. AMOD
dep.NSUBJ
dep.DET

Type system adoption / extension

- Adopted DKPro type system
  - Generic, well-designed type system with language independence in mind.
  - Granted EOP to use existing AEs already wrapped by DKPro.
- Then, we added some annotation types that were missing in DKPro
  - Semantic Role Labels, Alignment types, Predicate Truth value annotations, etc.
- Defined some types for T-H pair
  - Pairs, expression of entailment decision, TE metadata, etc.

Wrapping of LAP: UIMA is transparent to users

- LAP has its own interface methods
  - Wraps UIMA runtime, or any AE running methods
  - Each pipeline support those methods.
- Why wrap UIMA with additional interface?
  - Minimize users learning curve
    - Top level user don't need to know anything about UIMA.
    - Support TE specific capabilities.
  - "***Parallel***" adoption: project participants can implement LAP without UIMA AE/AAE adoption.
  - Cost of migration: "*Translating*" existing pipeline outputs to CAS is easier than break/migrate every components to AE

LAP Interface

- All LAP pipelines support a set of common functionalities (with Java API)
  - *generate an annotated T-H*, from string T-H pair.
  - *process RTE input file*, and generate a set of CASes.
  - *annotate* a given CAS.
- AE (Analysis Engine) based components
  - We recommend AE implementation for project members.
  - There is a common implementation that gets list of AEs, forms a pipeline, and automatically supports those common functionalities.

In the long term, we hope to get UIMA AE-based LAP components.

- Parallel adoption is an intermediate solution
  - “CAS only” adoption.
  - We hope this “parallel” adoption finally leads to all project members to adopt UIMA AE.
- For pluggable LAP components
  - New annotators are expected to have big impacts on various TE systems.
    - e.g. “Negation annotator”, “Predicate truth value annotator”
  - Without UIMA AE adoption, the user has to adopt the whole pipeline, not only the new module.

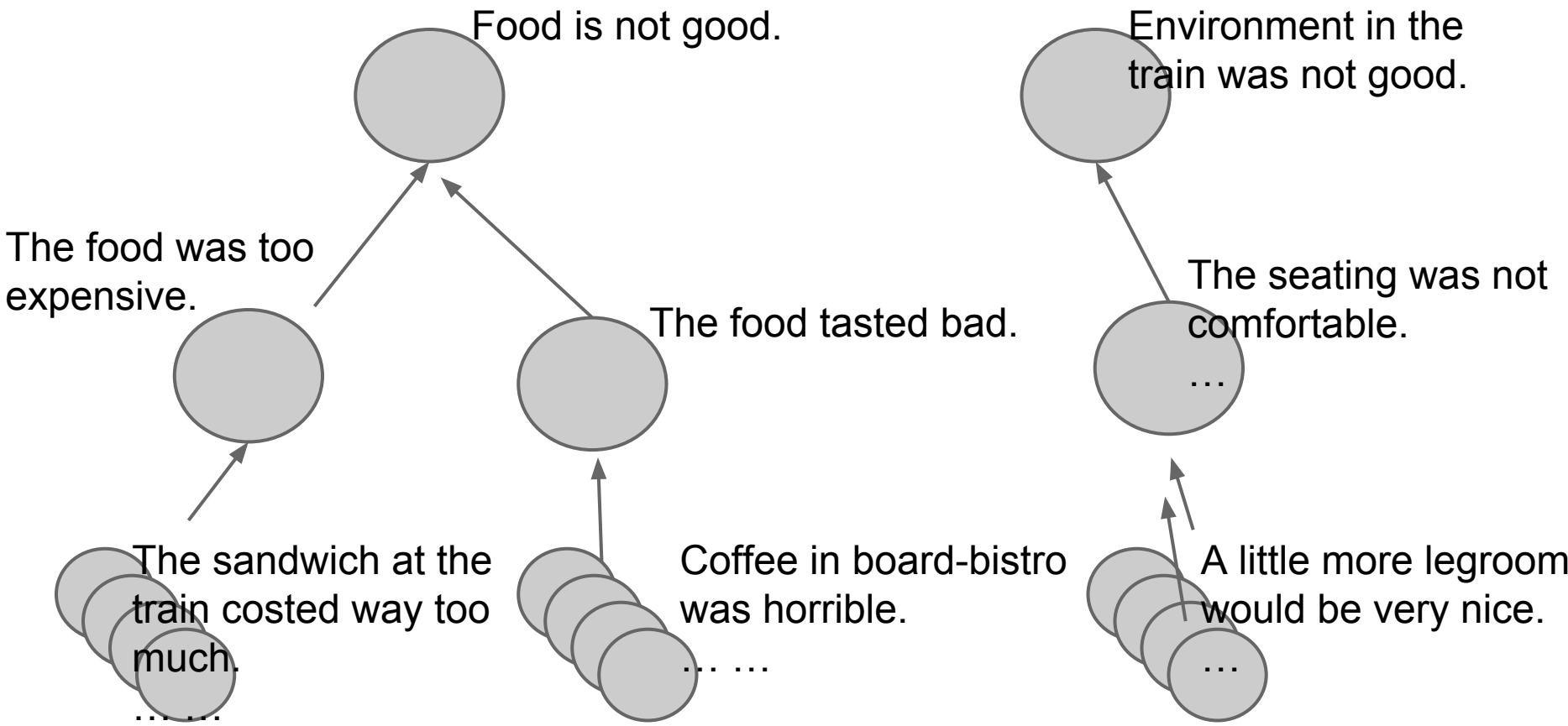
Currently -

- EOP Version 1.0 released in September 1.
- LAP
  - More than a dozen pipelines for 3 languages.
  - English, German, and Italian.
    - supports various levels of annotations
    - adoption of UIMA enables us to use existing AEs with low costs.
- CORE
  - Three systems have been migrated: TIE, EDITS, BIUTEE.
  - Working for English, German and Italian.
  - Various knowledge resources for the three languages

Open Issue #1: CAS in non-UIMA environment

- CAS is the object that holds all “annotated” data in EXCITEMENT platform.
- Widely used: even in some very complex data types!
  - Entailment Graph example
- CAS usage & Efficiency
  - UIMA recommends that minimize number of CASes.
  - But it is very easy for the platform users to treat CAS as “simply a data type that holds annotated data”. And use it as … just as a class.
  - **Lower Efficiency!**
  - Best practice needed, with better ways to store them

Entailment Graph Example

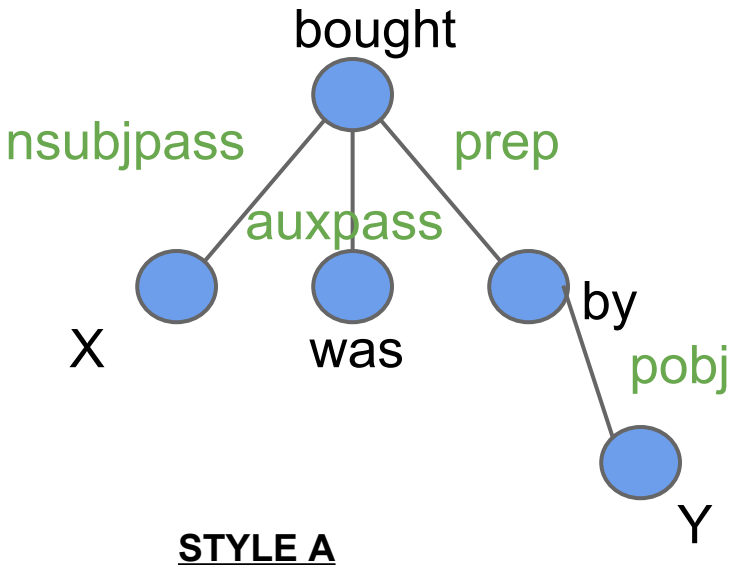


Open Issue #2: Annotation style "same parse tree in different style"

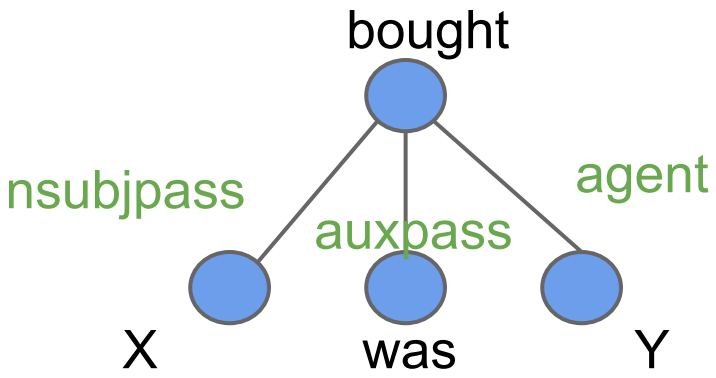
- Pluggable LAP
  - The goal is to make LAP independent from CORE; and LAP as replaceable. So if we get a new & better analyzer (e.g. parser), we can use that.
    - With a trivial re-training of core engine.
- However, some core components are depending on LAP output
  - Notably, parser and syntactic knowledge.
  - Parsers have "styles": knowledge components are affected by parsing output style.

Example: syntactic rule & different parse style

- Assume that we have one syntactic rule
  - ***X** was bought by **Y*** --entails→ ***Y** have **X***
- Different parse style example
  - Match would fail!



STYLE A



STYLE B

Open Issue #2: Annotation style and dependency

- Dependency between parser - syntactic knowledge.
  - A parser change will reduce the performance of knowledge resource, if they have different style.
- How bad is this?
  - Currently under investigation.
  - “Automatic parser style conversion” possible?
    - Automatically learning of conversion rules from two parsed corpora, etc.
  - Transform might be easier (or cheaper) than “re-generate” all knowledge resource.
  - “Self-contained” syntactic knowledge seems to be

Conclusion

- UIMA adoption enabled the project to have a good linguistic analysis pipeline.
  - Multilingual, metadata-rich linguistic analysis pipeline.
- Existing work of the community helped us to build various pipelines with ease.
  - DKPro type systems and its AEs.
- In the project, CAS is the standard data representation for annotated data
  - CAS can be passed and used successfully in non-UIMA environment.

Thanks!

- EXCITEMENT open platform 1.0
  - You can try it by visiting the following URL.

http://hltfbk.github.io/Excitement-Open-Platform/

- NOTE: Still in a testing phase.