COGNITIVE COMPUTATION GROUP
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

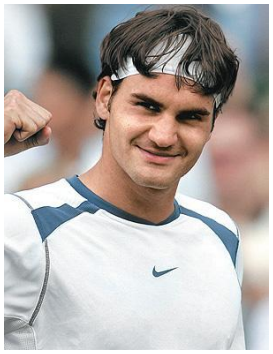
An Introduction to Machine Learning and Natural Language Processing Tools

MIAS
MULTIMODAL INFORMATION ACCESS AND SYNTHESIS

Vivek Srikumar, Mark Sammons
(Some slides from Nick Rizzolo)

The Famous People Classifier

f() = Politician

f() = Athlete

f() = Corporate Mogul

Outline

- An Overview of NLP Resources
- Our NLP Application: The Fame classifier
- The Curator
- Edison
- Learning Based Java
- Putting everything together

What can NLP do for me?

An overview of NLP resources

NLP resources (An incomplete list)

- Cognitive Computation Group resources
  - Tokenization/Sentence Splitting
  - Part Of Speech
  - Chunking
  - Named Entity Recognition
  - Coreference
  - Semantic Role Labeling

- Others
  - Stanford parser and dependencies
  - Charniak Parser

Tokenization and Sentence Segmentation

- Given a document, find the sentence and token boundaries

  The police chased Mr. Smith of Pink Forest, Fla. all the way to Bethesda, where he lived. Smith had escaped after a shoot-out at his workplace, Machinery Inc.

- Why?
  - Word counts may be important features
  - Words may themselves be the object you want to classify
  - "lived." and "lived" should give the same information
  - different analyses need to align if you want to leverage multiple annotators from different sources/tasks

Part of Speech (POS)

- Allows simple abstraction for pattern detection

POS	DT	NN	VBD	PP	DT	JJ	NN
Word	The	boy	stood	on	the	burning	deck

POS	DT	NN	VBD	PP	DT	JJ	NN
Word	A	boy	rode	on	a	red	bicycle

- Disambiguate a target, e.g.

  "make (a cake)" vs. "make (of car)"

- Specify more abstract patterns,

  e.g. Noun Phrase: (DT JJ* NN)

- Specify context in abstract way
  - e.g. "DT boy VBX" for "actions boys do"
  - This expression will catch "a boy cried", "some boy ran", …

Chunking

- Identifies phrase-level constituents in sentences

  [NP Boris] [ADVP regretfully] [VP told] [NP his wife] [SBAR that] [NP their child] [VP could not attend] [NP night school] [PP without] [NP permission] .

- Useful for filtering: identify e.g. only noun phrases, or only verb phrases
  - Groups modifiers with heads
  - Useful for e.g. Mention Detection
- Used as source of features, e.g. distance (abstracts away determiners, adjectives, for example), sequence,…
  - More efficient to compute than full syntactic parse
  - Applications in e.g. Information Extraction – getting (simple) information about concepts of interest from text documents

Named Entity Recognition

- Identifies and classifies strings of characters representing proper nouns

  [PER Neil A. Armstrong] , the 38-year-old civilian commander, radioed to earth and the mission control room here: "**[LOC Houston]** , **[ORG Tranquility]** Base here; the Eagle has landed."

- Useful for filtering documents
  - "I need to find news articles about organizations in which Bill Gates might be involved…"
- Disambiguate tokens: "Chicago" (team) vs. "Chicago" (city)
- Source of abstract features
  - E.g. "Verbs that appear with entities that are Organizations"
  - E.g. "Documents that have a high proportion of Organizations"

Coreference

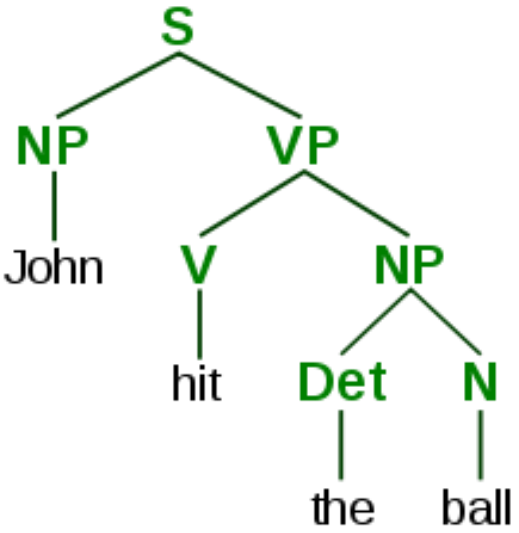
- Identify all phrases that refer to each entity of interest – i.e., group mentions of concepts

  [Neil A. Armstrong] , **[the 38-year-old civilian commander]**, radioed to **[earth]**. **[He]** said the famous words, "**[the Eagle]** has landed"."

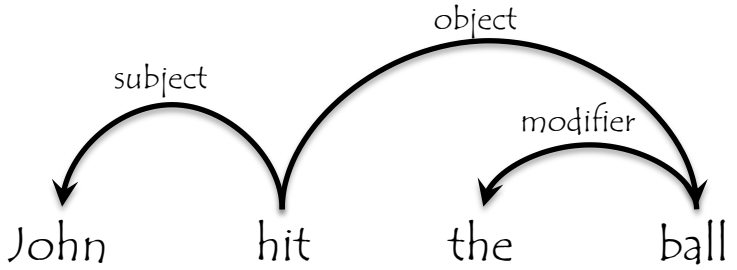
- The Named Entity recognizer only gets us part-way…
- …if we ask, "what actions did Neil Armstrong perform?", we will miss many instances (e.g. "He said…")
- Coreference resolver abstracts over different ways of referring to the same person
  - Useful in feature extraction, information extraction

Parsers

- Identify the grammatical structure of a sentence



Full parse



Dependency parse

Parsers reveal the grammatical relationships between words and phrases

Semantic Role Labeler

Semantic Role Labeling Output

Input Text:

A car bomb that exploded outside the U.S. military base in Beniji killed 11 Iraqi citizens.

Result: Complete!

⊞ General Explanation of Argument Labels

A	bomb [A1]		killer [A0]
car			
bomb			
that	bomb (Reference) [R-A1]		
exploded	V: explode		
outside	location [AM-LOC]		
the			
U.S.			
military	temporal [AM-TMP]		
base			
in	location [AM-LOC]		
Beniji			
killed			V: kill
11			corpse [A1]
Iraqi			
citizens			
.			

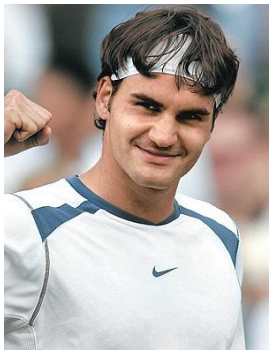
- SRL reveals relations and arguments in the sentence (where relations are expressed as verbs)
- Cannot abstract over variability of expressing the relations – e.g. kill vs. murder vs. slay…

Enough NLP. Let's make our $$$ with the

The fame classifier

The Famous People Classifier

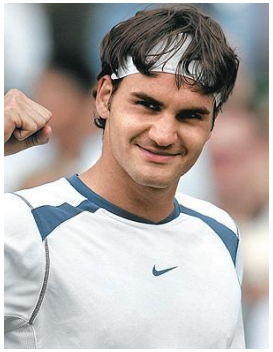
f() = Politician

f() = Athlete

f() = Corporate Mogul

The NLP version of the fame classifier

All sentences in the news, which the string **Barack Obama** occurs



Represented by

All sentences in the news, which the string **Roger Federer** occurs

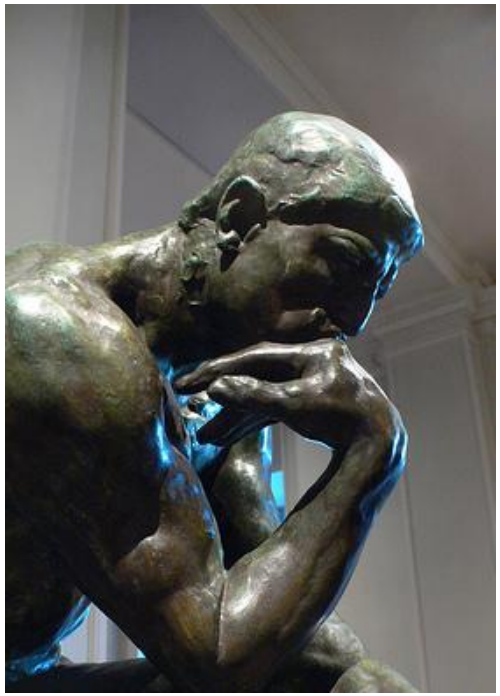
All sentences in the news, which the string **Bill Gates** occurs

Our goal

- Find famous athletes, corporate moguls and politicians

Athlete	Politician	Corporate Mogul
◇ Michael Schumacher ◇ Michael Jordan ◇ …	◇ Bill Clinton ◇ George W. Bush ◇ …	◇ Warren Buffet ◇ Larry Ellison ◇ …

Let's brainstorm



- What NLP resources could we use for this task?

  Remember, we start off with just raw text from a news website

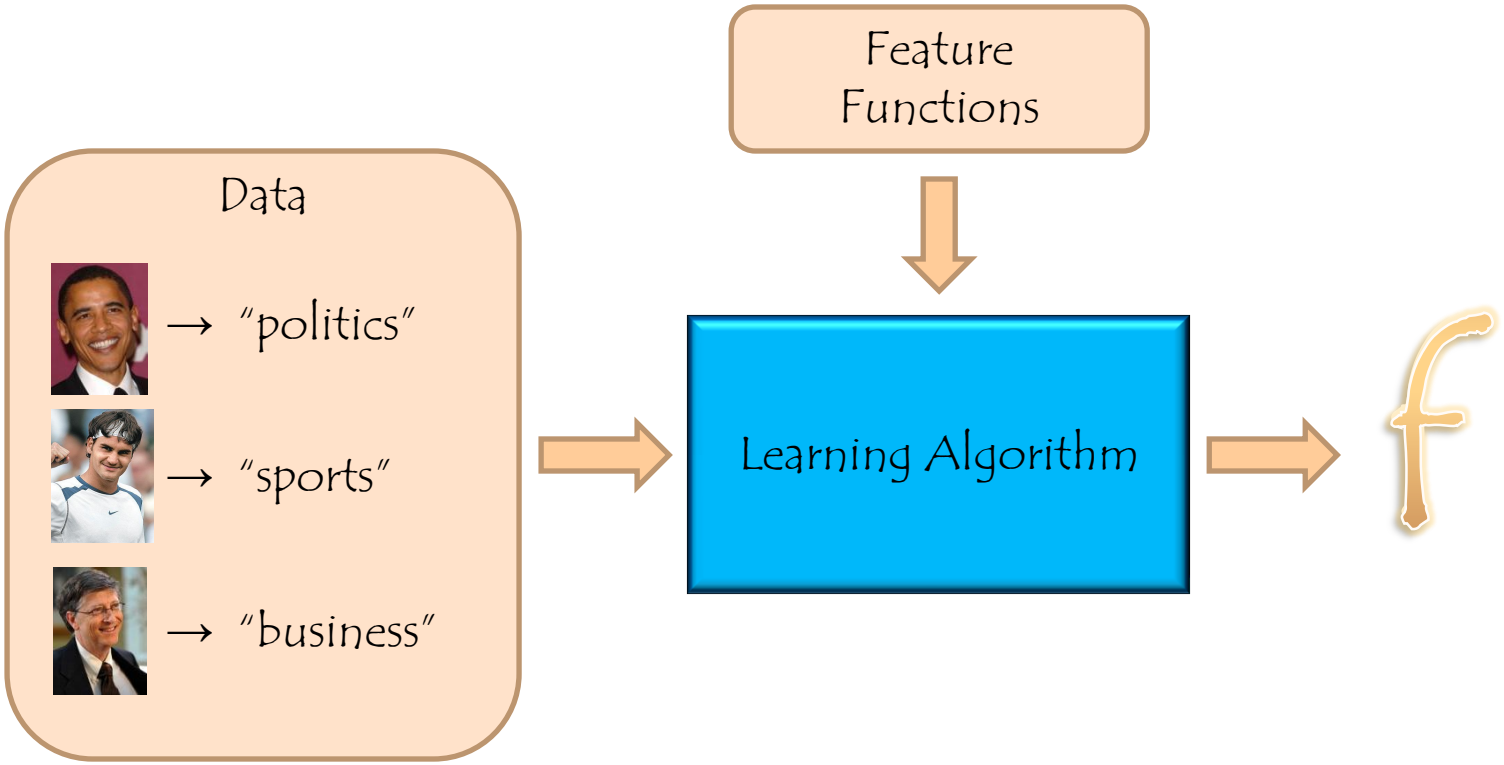
One solution

- Let us label entities using features defined on mentions

All sentences in the news, which the string **Barack Obama** occurs

- Identify mentions using the named entity recognizer
- Define features based on the words, parts of speech and dependency trees
- Train a classifier

Where to get it: Machine Learning



A second look at the solution

- Identify mentions using the named entity recognizer
- Define features based on the words, parts of speech and dependency trees
- Train a classifier

University of Illinois

Sentence and Word Splitter
Part-of-speech Tagger
Named Entity Recognizer

Stanford University

Dependency Parser
(and the NLP pipeline)

These tools can be downloaded from the websites.
Are we done? If not, what's missing?

We need to put the pieces together

The infrastructure

The Curator

- A common interface for different NLP annotators
- Caches their results

Edison

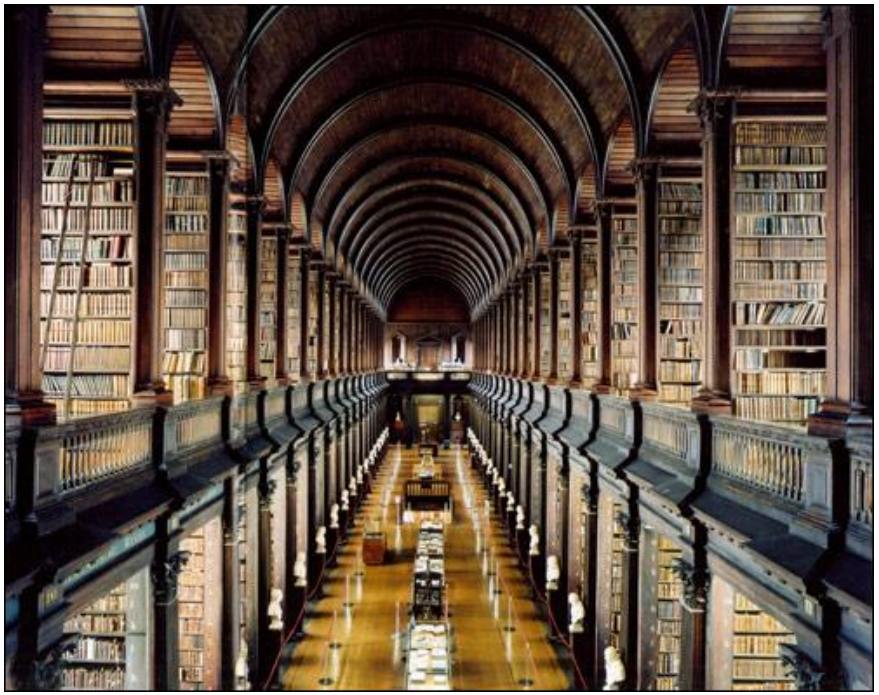
- Library for NLP representation in Java
- Helps with extracting complex features

Learning Based Java

- A Java library for machine learning
- Provides a simple language to define classifiers and perform inference with them

The infrastructure

- Each infrastructure module has specific interfaces that the user is expected to use
- The Curator specifies the interface for accessing annotations from the NLP tools
- Edison fixes the representation for the NLP annotation
- Learning Based Java requires training data to be presented to it using an interface called **Parser**



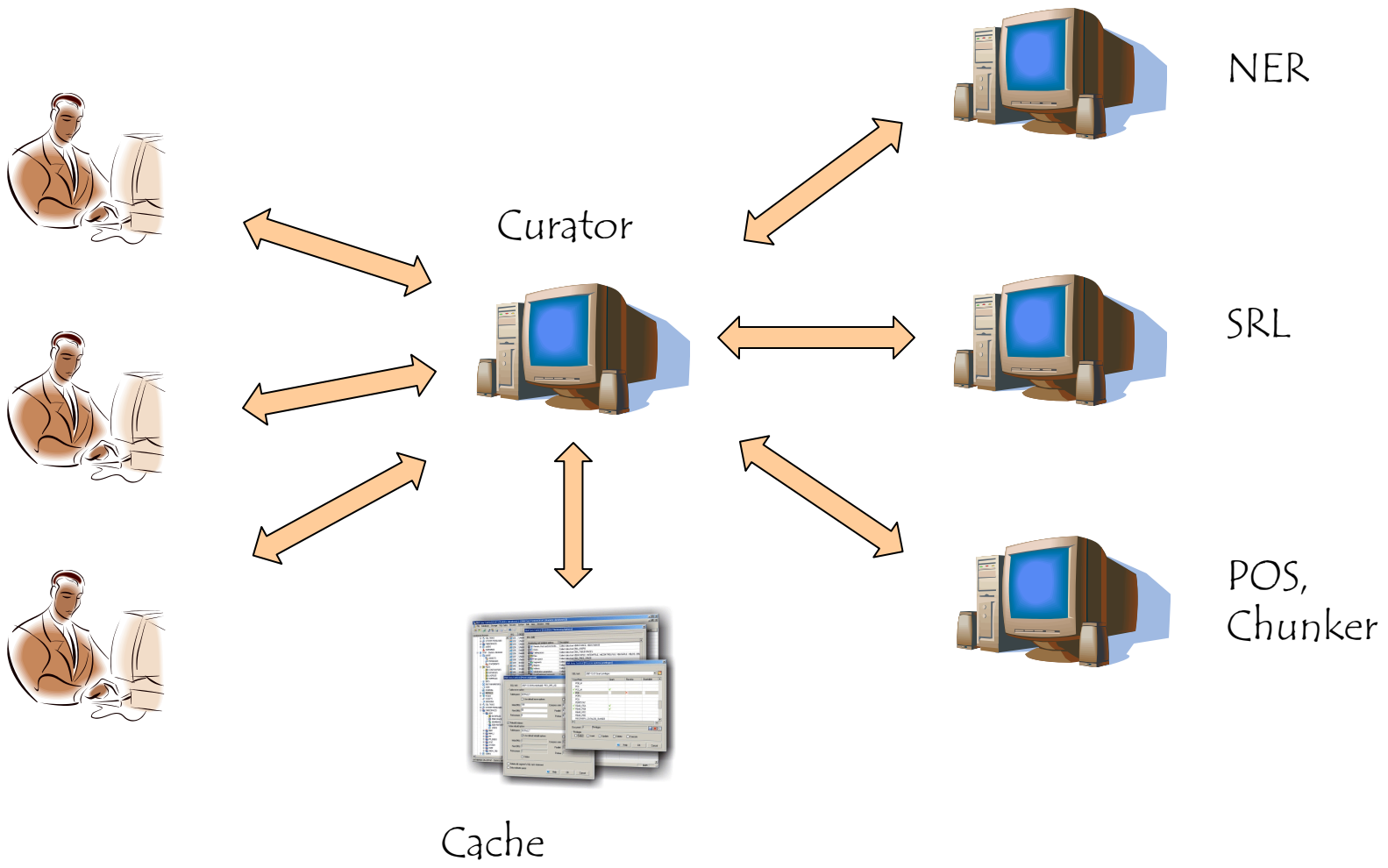
A place where NLP annotations live

Curator

Big NLP

- NLP tools are quite sophisticated
- The more complex, the bigger the memory requirement
  - NER: 1G; Coref: 1G; SRL: 4G ….
- If you use tools from different sources, they may be…
  - In different languages
  - Using different data structures
- If you run a lot of experiments on a single corpus, it would be nice to cache the results
  - …and for your colleagues, nice if they can access that cache.
- Curator is our solution to these problems.

Curator



What does the Curator give you?

- Supports distributed NLP resources
  - Central point of contact
  - Single set of interfaces
  - Code generation in many programming languages (using Thrift)
- Programmatic interface
  - Defines set of common data structures used for interaction
- Caches processed data
- Enables highly configurable NLP pipeline

Overhead:

- Annotation is all at the level of character offsets: Normalization/mapping to token level required
- Need to wrap tools to provide requisite data structures

Getting Started With the Curator

http://cogcomp.cs.illinois.edu/curator

- Installation:
  - Download the curator package and uncompress the archive
  - Run **bootstrap.sh**

- The default installation comes with the following annotators (Illinois, unless mentioned):
  - Sentence splitter and tokenizer
  - POS tagger
  - Shallow Parser
  - Named Entity Recognizer
  - Coreference resolution system
  - Stanford parser

Basic Concept

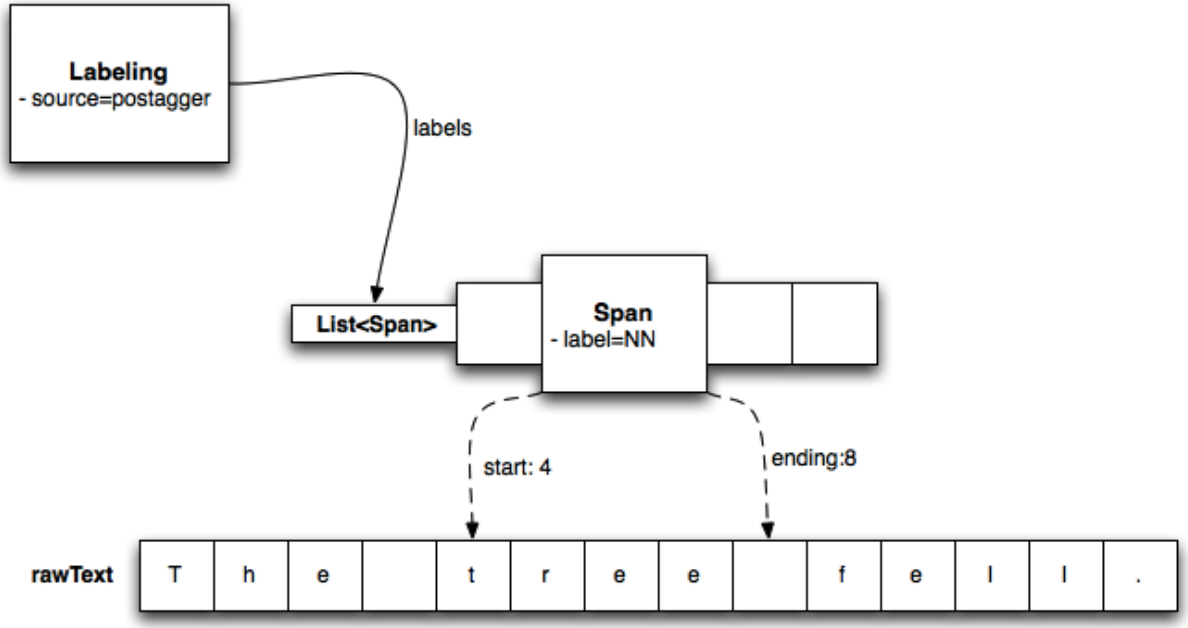
- Different NLP annotations can be defined in terms of a few simple data structures:
  1. **Record:** A big container to store all annotations of a text
  2. **Span**: A span of text (defined in terms of characters) along with a label (A single token, or a single POS tag)
  3. **Labeling**: A collection of **Span**s (POS tags for the text)
  4. **Trees** and **Forests** (Parse trees)
  5. **Clustering**: A collection of **Labeling**s (Co-reference)

Go here for more information:

http://cogcomp.cs.illinois.edu/trac/wiki/CuratorDataStructures

Example of a Labeling

The tree fell.



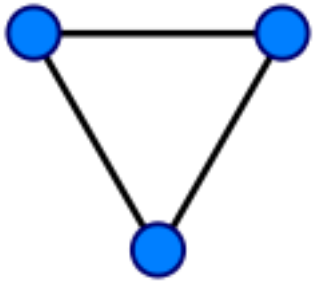
Representing NLP objects and extracting features

Edison

Edison

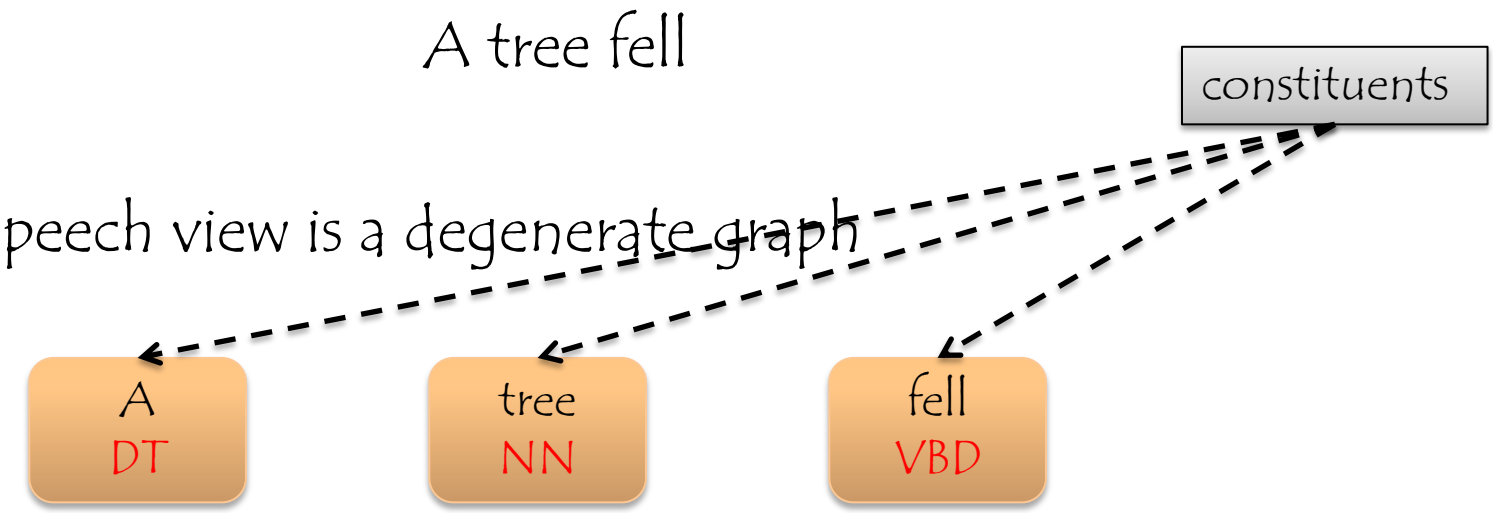
- An NLP data representation and feature extraction library
- Helps manage and use different annotations of text
- Doesn't the Curator do everything we need?
  - Curator is a service that abstracts away different annotators
  - Edison is a Curator client
  - And more…

Representation of NLP annotations



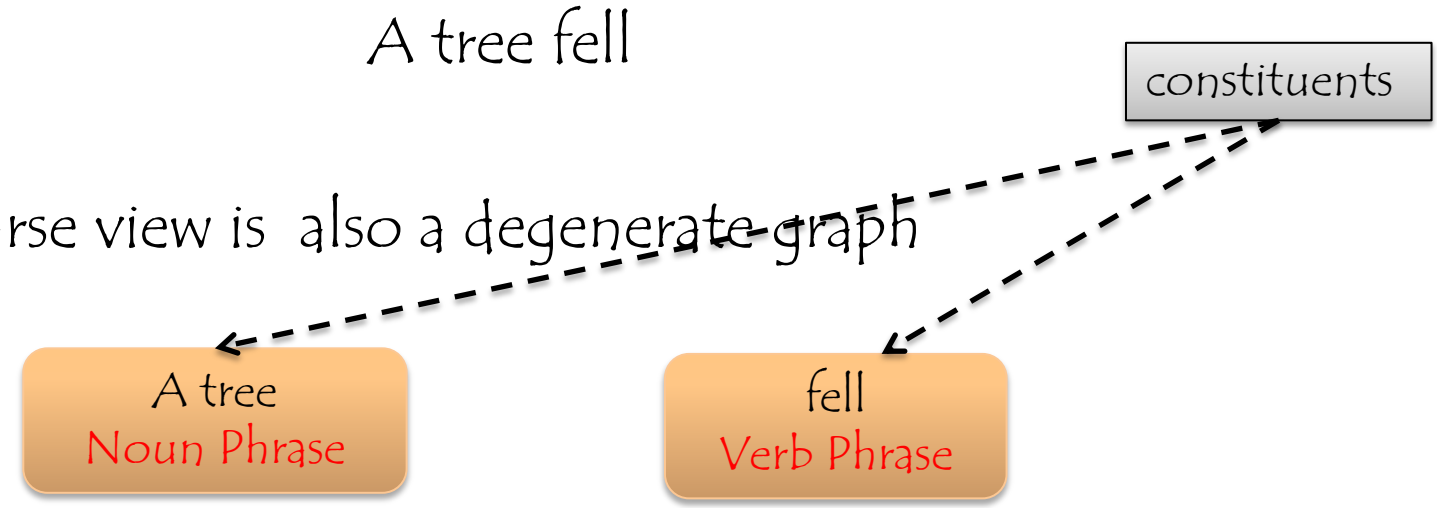
- All NLP annotations are called **Views**
- A View is just a labeled directed graph
  - Nodes are labeled collections of tokens, called **Constituents**
  - Labeled edges between nodes are called **Relations**
- All Views related to some text are contained in a **TextAnnotation**

Example of Views: Part of speech



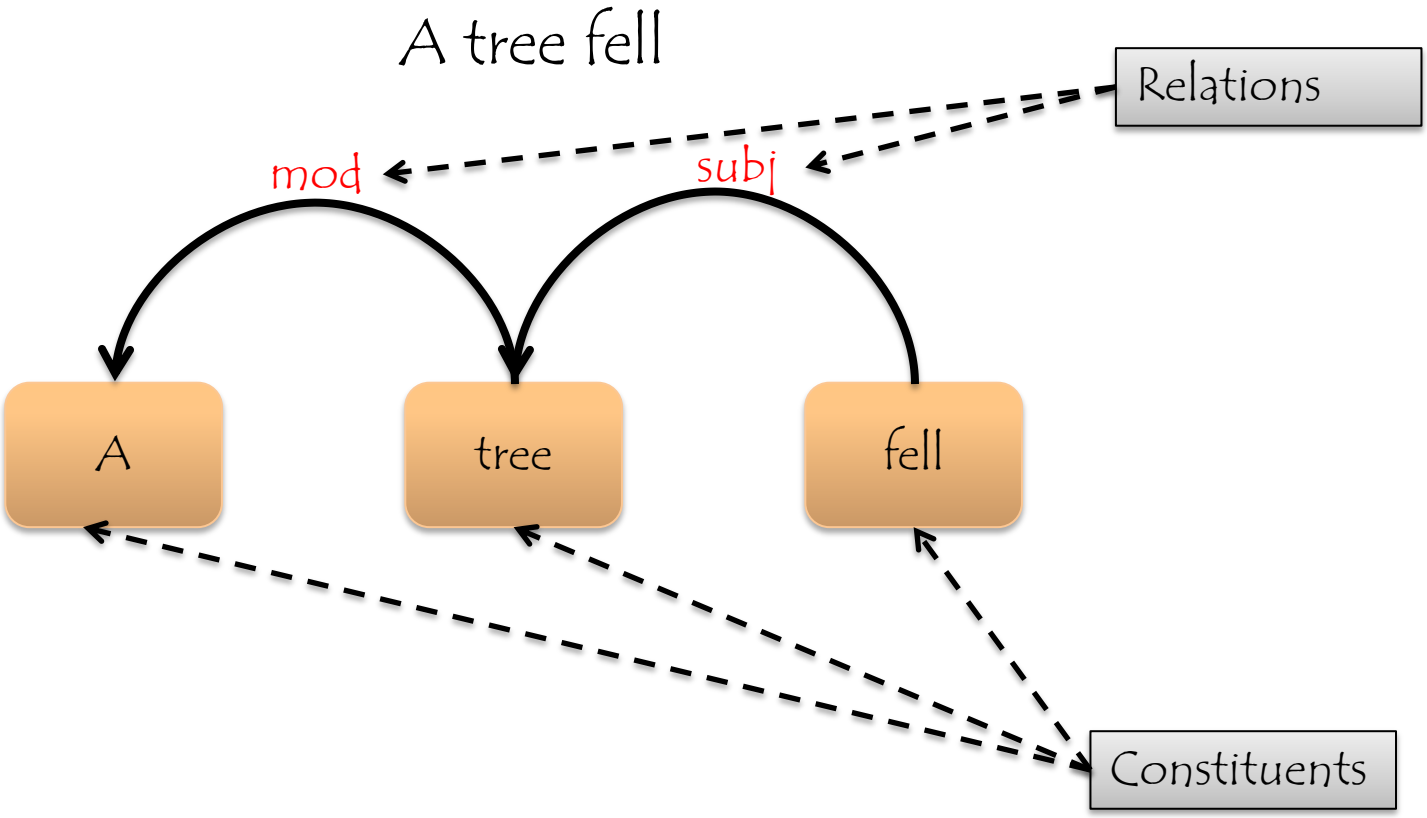
- Part of speech view is a degenerate graph
  - No edges because there are no relations
  - This kind of View is represented by a subclass called TokenLabelView
- Note that constituents are token based, not character based

Example of Views: Shallow Parse



- Shallow parse view is also a degenerate graph
  - No edges because there are no relations
  - This kind of View is represented by a subclass called SpanLabelView

Example of Views: DependencyTree



- A subclass of View called **TreeView**

More about Views

- **View** represents a generic graph of Constituents and Relations

- Its subclasses denote specializations suited to specific structures
  - TokenLabelView
  - SpanLabelView
  - TreeView
  - PredicateArgumentView
  - CoreferenceView

- Each view allows us to query its constituents
  - Useful for defining features!

Features

- Complex features using this library
- Examples
  - POS tag for a token
  - All POS tags within a span
  - All tokens within a span that have a specific POS tag
  - All chunks contained within a parse constituent
  - All chunks contained in the largest NP that covers a token
  - All co-referring mentions to chunks contained in the largest NP that covers this token
  - All incoming dependency edges to a constituent
- Enables quick feature engineering

Getting started with Edison

http://cogcomp.cs.uiuc.edu/software/edison

- How to use Edison:
  1. Download the latest version of Edison and its dependencies from the website
  2. Add all the jars to your project
  3. ????
  4. Profit

- A Maven repository is also available. See the edison page for more details

Demo 1

- Basic Edison example, where we will

1. Create a TextAnnotation object from raw text
2. Add a few views from the curator
3. Print them on the terminal

http://cogcomp.cs.uiuc.edu/software/edison/FirstCuratorExample.html

Demo 2

- Second Edison example, where we will

1. Create a TextAnnotation object from raw text
2. Add a few views from the curator
3. Print all the constituents in the named entity view

Let's recall our goal

- Let us label entities using features defined on mentions

All sentences in the news, which the string **Barack Obama** occurs

- Identify mentions using the named entity recognizer
- Define features based on the words, parts of speech and dependency trees
- Train a classifier

Demo 3

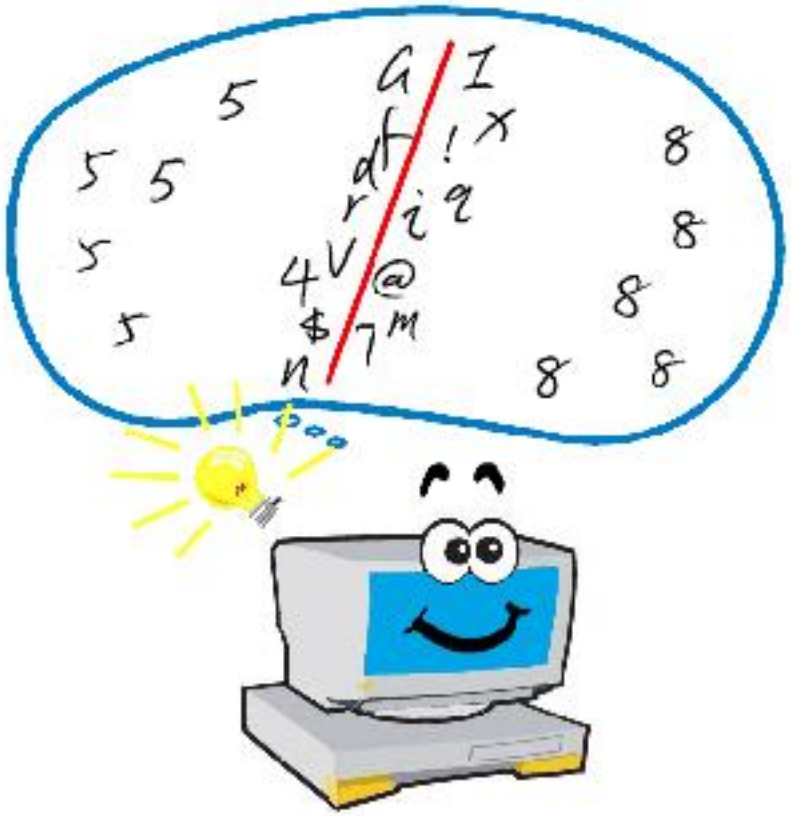
- Reading the Fame classifier data and adding views
- Feature functions
  - What would be good features for the fame classification task?

The US President **Barack Obama** said that he ….

President **Barack Obama** recently visited France.

Features for **Barack Obama**

- US: 1
- President: 2
- said: 1
- visited: 1
- France: 1



Writing classifiers

Learning Based Java

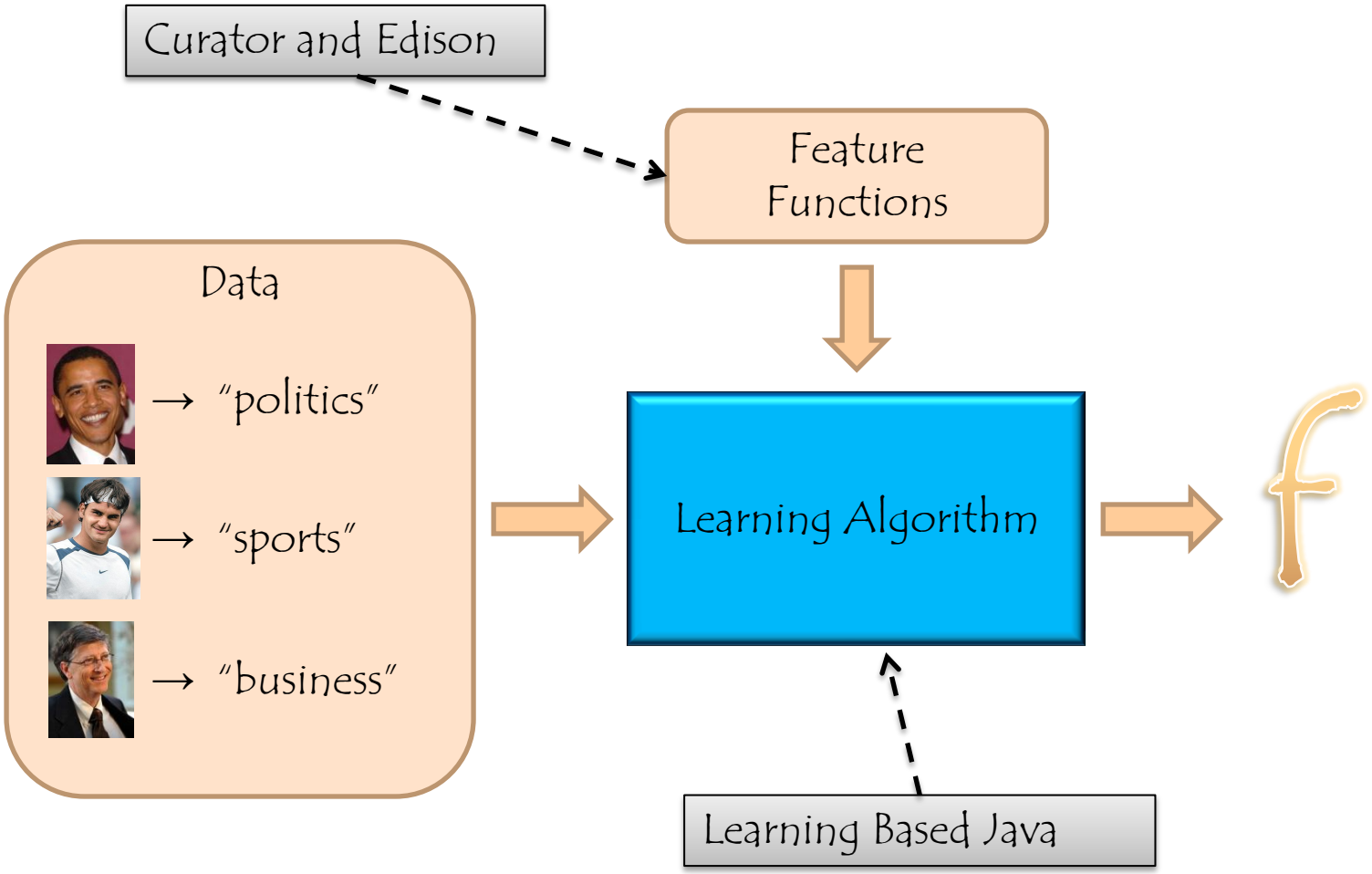
What is Learning Based Java?

- A modeling language for learning and inference
- Supports
  - Programming using learned models
  - High level specification of features and constraints between classifiers
  - Inference with constraints
- The learning operator
  - Classifiers are functions defined in terms of data
  - Learning happens at *compile time*

What does LBJ do for you?

- Abstracts away the feature representation, learning and inference
- Allows you to write *learning based programs*
- Application developers can reason about the application at hand

Our application



Demo 4

- The fame classifier itself
  1. The features
  2. The classifier
  3. Compiling to train the classifier

Putting the pieces together

The Fame classifier

Recall our solution

- Let us label entities using features defined on mentions

All sentences in the news, which the string **Barack Obama** occurs

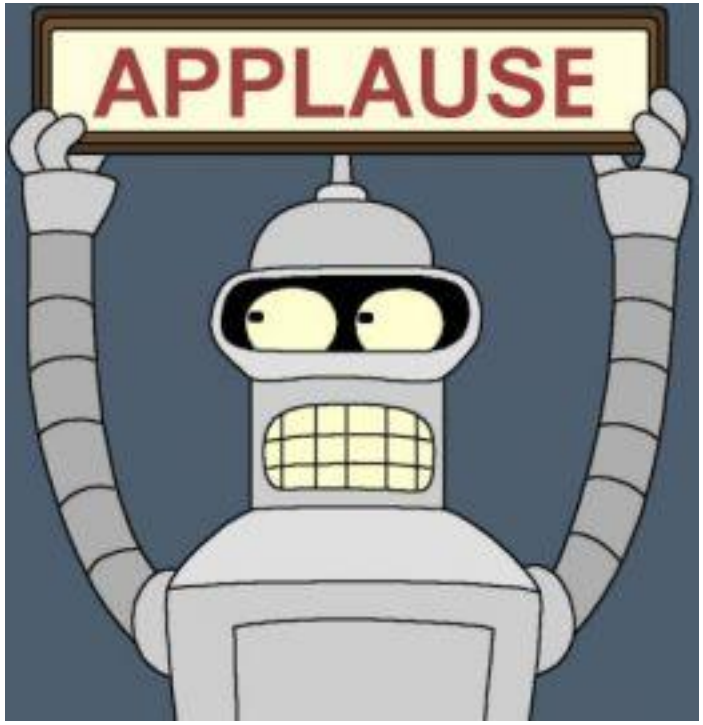
- Identify mentions using the named entity recognizer
- Define features based on the words, parts of speech and dependency trees
- Train a classifier

The infrastructure

- Curator
  - Provides access to the POS tagger, NER and the Stanford Dependency parser
  - Caches all annotations
- Edison
  - NLP representation in our program
  - Feature extraction
- Learning Based Java
  - The machine learning

Final demo

- Let's see this in action



Links

- Cogcomp Software:

  http://cogcomp.cs.illinois.edu/page/software

- Support:

  illinois-ml-nlp-users@cs.uiuc.edu

- Download the slides and the code from

  http://cogcomp.cs.illinois.edu/page/tutorial.201008

Running the test code on a Unix Machine

Step 1: Train the classifier

```
$ ./compileLBJ entityFame.lbj
```

Step 2: Compile the other java files with

```
$ ant
```

Step 3: Test the classifier:

```
$ ./test.sh data/test
```