Domain Adaptive IE: Learning Template Filling Rules

Günter Neumann & Feiyu Xu

{neumann, feiyu}@dfki.de

Language Technology-Lab

DFKI, Saarbrücken

Motivations

- Porting to new domains or applications is expensive
- Current technology requires IE experts
  - Expertise difficult to find on the market
  - SME cannot afford IE experts
- Machine learning approaches
  - Domain portability is relatively straightforward
  - System expertise is not required for customization
  - “Data driven” rule acquisition ensures full coverage of examples

Problems

- Training data may not exist, and may be very expensive to acquire
- Large volume of training data may be required
- Changes to specifications may require reannotation of large quantities of training data
- Understanding and control of a domain adaptive system is not always easy for non-experts

Parameters

- Document structure
  - Free text
  - Semi-structured
  - Structured
- Richness of the annotation
  - Shallow NLP
  - Deep NLP
- Complexity of the template filling rules
  - Single slot
  - Multi slot
- Amount of data

- Degree of automation
  - Semi-automatic
  - Supervised
  - Semi-Supervised
  - Unsupervised
- Human interaction/contribution
- Evaluation/validation
  - during learning loop
  - Performance: recall and precision

Learning Methods for Template Filling Rules

- Inductive learning
- Statistical methods
- Bootstrapping techniques
- Active learning

Documents

- Unstructured (Free) Text
  - Regular sentences and paragraphs
  - Linguistic techniques, e.g., NLP
- Structured Text
  - Itemized information
  - Uniform syntactic clues, e.g., table understanding
- Semi-structured Text
  - Ungrammatical, telegraphic (e.g., missing attributes, multi-value attributes, …)
  - Specialized programs, e.g., wrappers

"Information Extraction" From Free Text

October 14, 2002, 4:00 a.m. PT

For years, Microsoft Corporation CEO Bill Gates railed against the economic philosophy of open-source software with Orwellian fervor, denouncing its communal licensing as a "cancer" that stifled technological innovation.

Today, Microsoft claims to "love" the open-source concept, by which software code is made public to encourage improvement and development by outside programmers. Gates himself says Microsoft will gladly disclose its crown jewels--the coveted code behind the Windows operating system--to select customers.

"We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access.“

Richard Stallman, founder of the Free Software Foundation, countered saying…

*

Microsoft Corporation
CEO
Bill Gates

*

Microsoft
Gates
Microsoft

*

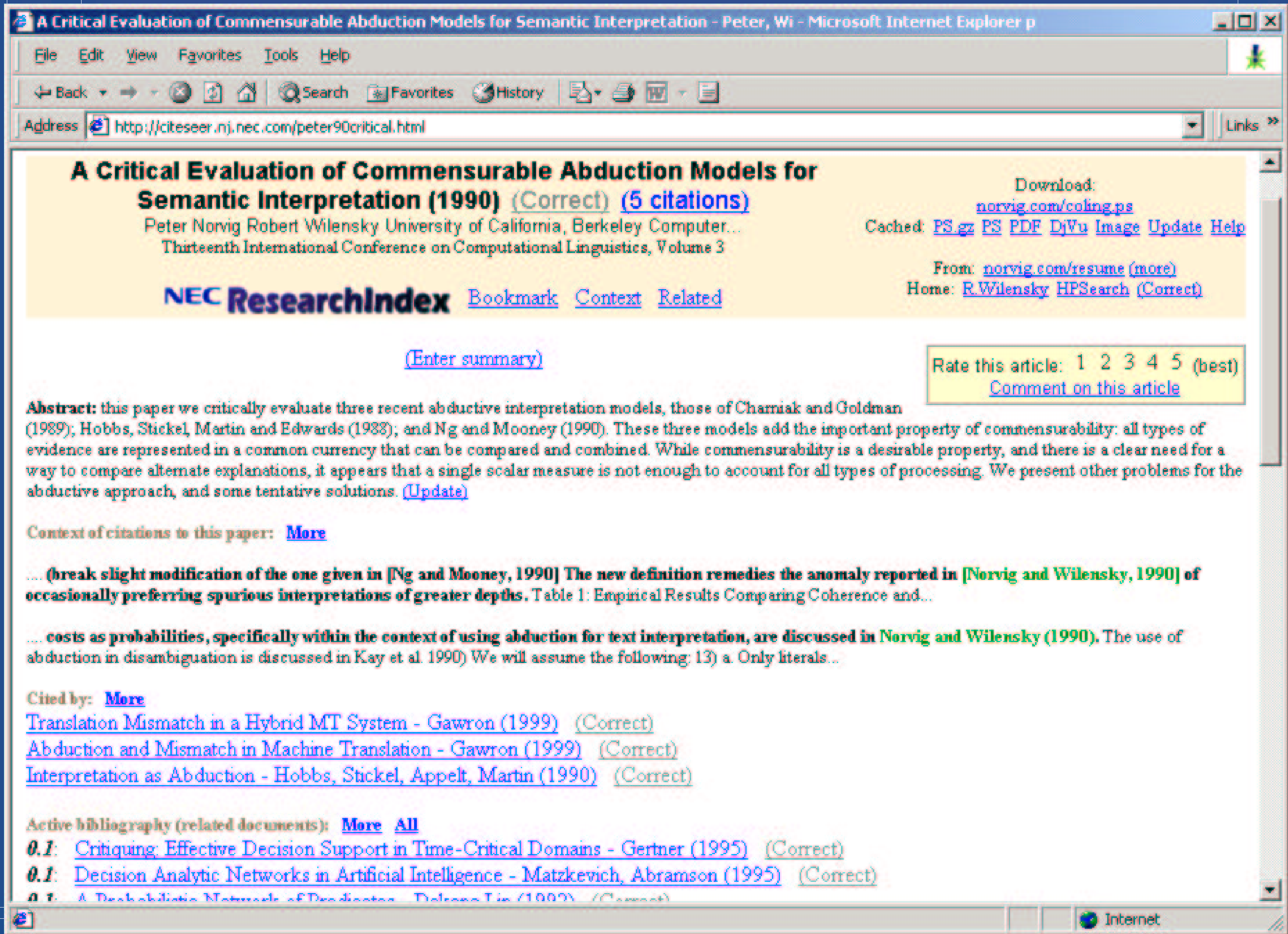
Bill Veghte
Microsoft
VP

*

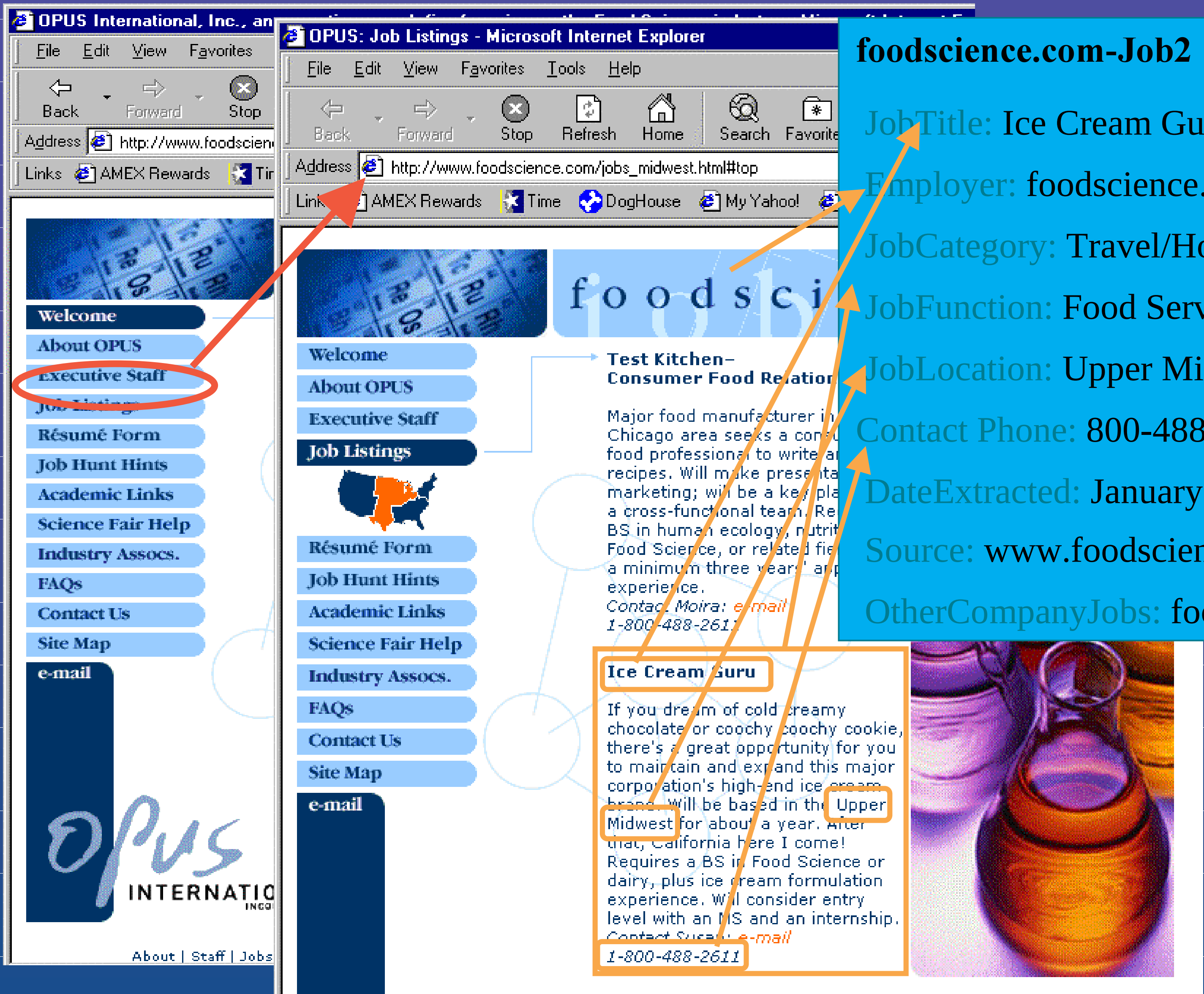
Richard Stallman
founder
Free Software Foundation

NAME	TITLE	ORGANIZATION
Bill Gates	CEO	Microsoft
Bill Veghte	VP	Microsoft
Richard Stallman	founder	Free Soft..

IE from Research Papers



Extracting Job Openings from the Web: Semi-Structured Data



foodscience.com-Job2

JobTitle: Ice Cream Guru

Employer: foodscience.com

JobCategory: Travel/Hospitality

JobFunction: Food Services

JobLocation: Upper Midwest

Contact Phone: 800-488-2611

DateExtracted: January 8, 2001

Source: www.foodscience.com/jobs_midwest.htm

OtherCompanyJobs: foodscience.com-Job1

Outline

- Free text
  - Supervised and semi-automatic
    - AutoSlog
  - Semi-Supervised
    - AutoSlog-TS
  - Unsupervised
    - ExDisco
- Semi-structured and unstructured text
  - NLP-based wrapping techniques
    - RAPIER

Free Text

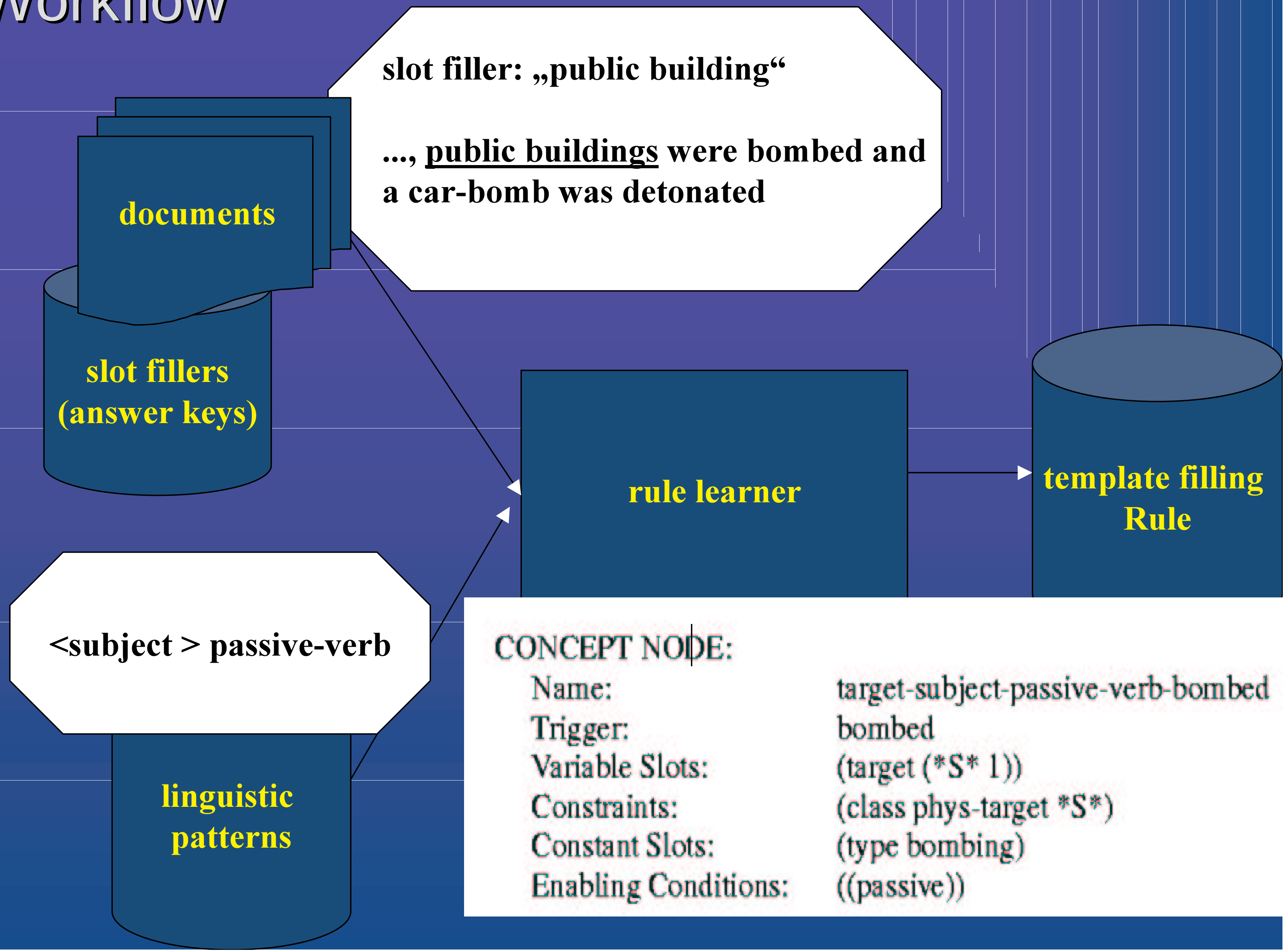
NLP-based Supervised Approaches

- Input is an annotated corpus
  - Documents with associated templates
- A parser
  - Chunk parser
  - Full sentence parser
- Learning the mapping rules
  - From linguistic constructions to template fillers

AutoSlog (1993)

- Extracting a concept dictionary for template filling
- Full sentence parser
- One slot filler rules
- Domain adaptation performance
  - Before AutoSlog: hand-crafted dictionary
    - two highly skilled graduate students
    - 1500 person-hours
  - AutoSlog:
    - A dictionary for the terrorist domain: 5 person hours
    - 98% performance achievement of the hand-crafted dictionary

Workflow



slot filler: „public building“
..., public buildings were bombed and a car-bomb was detonated
documents
slot fillers (answer keys)
rule learner
template filling Rule
<subject > passive-verb
linguistic patterns
CONCEPT NODE:
Name: target-subject-passive-verb-bombed
Trigger: bombed
Variable Slots: (target (*S* 1))
Constraints: (class phys-target *S*)
Constant Slots: (type bombing)
Enabling Conditions: ((passive))

Linguistic Patterns

Linguistic Pattern	**Example**
<subject> passive-verb	<victim> was murdered
<subject> active-verb	<perpetrator> bombed
<subject> verb infinitive	<perpetrator> attempted to kill
<subject> auxiliary noun	<victim> was victim
passive-verb <dobj>[1]	killed <victim>
active-verb <dobj>	bombed <target>
infinitive <dobj>	to kill <victim>
verb infinitive <dobj>	threatened to attack <target>
gerund <dobj>	killing <victim>
noun auxiliary <dobj>	fatality was <victim>
noun prep <np>	bomb against <target>
active-verb prep <np>	killed with <instrument>

Id: DEV-MUC4-1192 **Slot filler:** “gilberto molasco”
Sentence: (they took 2-year-old gilberto molasco, son of patricio rodriguez, and 17-year-old andres argueta, son of emimesto argueta.)

CONCEPT NODE

Name:	victim-active-verb-dobj-took
Trigger:	took
Variable Slots:	(victim (*DOBJ* 1))
Constraints:	(class victim *DOBJ*)
Constant Slots:	(type kidnapping)
Enabling Conditions:	((active))

[redacted] A bad concept node definition

Error Sources

- A sentence contains the answer key string but does not contain the event
- The sentence parser delivers wrong results
- A heuristic proposes a wrong conceptual anchor

Training Data

- MUC-4 corpus
- 1500 texts
- 1258 answer keys
- 4780 string fillers
- 1237 concept node definition

- Human in loop for validation to filter out bad and wrong definitions: 5 hours

- 450 concept nodes left after human review

System/Test Set	Recall	Precision	F-measure
MUC-4/TST3	46	56	50.51
AutoSlog/TST3	43	56	48.65
MUC-4/TST4	44	40	41.90
AutoSlog/TST4	39	45	41.79

[redacted] Comparative Results

Summary

- Advantages
  - Semi-automatic
  - Less human effort
- Disadvantages
  - Human interaction
  - Still very naive approach
  - Need a big amount of annotation
  - Domain adaptation bottelneck is shifted to human annotation
  - No generation of rules
  - One slot filling rule
  - No mechanism for filtering out bad rules

NLP-based ML Approaches

- LIEP (Huffman, 1995)
- PALKA (Kim & Moldovan, 1995)
- HASTEN (Krupka, 1995)
- CRYSTAL (Soderland et al., 1995)

CRYSTAL [1995]

The Parliament building was bombed by ***Carlos***.

Concept type: BUILDING BOMBING

SUBJECT:	Classes include:	<PhysicalTarget>
	Terms include:	BUILDING
	Extract:	*target*
VERB:	Root:	BOMB
	Mode:	passive
PREPOS-PHRASE:	Preposition:	BY
	Classes include:	<PersonName>
	Extract:	*perpetrator name*

PALKA [1995]

The Parliament building was bombed by ***Carlos***.

FP-structure = MeaningFrame + PhrasalPattern

```
Meaning Frame:   (BOMBING agent:      ANIMATE
                          target:     PHYS-OBJ
                          instrument: PHYS-OBJ
                          effect:     STATE)

Phrasal Pattern:   ((PHYS-OBJ) was bombed by (PERP))

FP-structure:
  (BOMBING  target:   PHYS-OBJ
            agent:    PERP
            pattern:  ((target) was bombed by (agent))
```

LIEP [1995]

The Parliament building was bombed by *Carlos*.

```
TARGET-was-bombed-by-PERPETRATOR:
  noun-group( TRGT, head( isa(physical-target) ) ),
  noun-group( PERP, head( isa(perpetrator) ) )
  verb-group( VG, type(passive), head(bombed) )
  preposition( PREP, head(by) )

  subject( TRGT, VG ),
  post-verbal-prep( VG, PREP ),
  prep-object( PREP, PERP )
  ⟹ bombing-event( BE, target(TRGT), agent(PERP) )
```

HASTEN [1995]

The Parliament building was bombed by *Carlos*.

BOMBING:

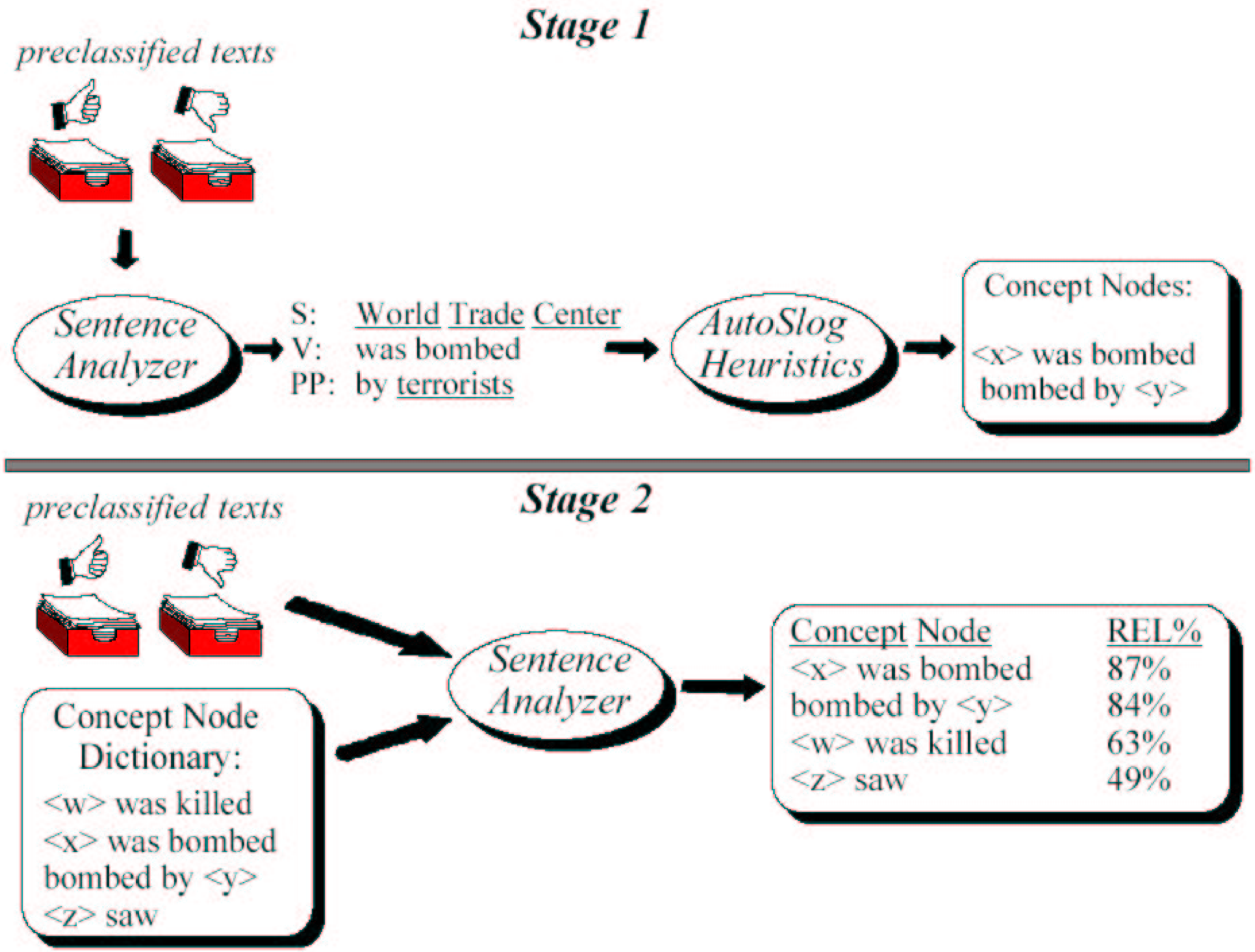
TARGET:	NP "semantic = physical-object"
ANCHOR:	VG "root = bomb"
PERPETRATOR:	NP "semantic = terrorist-group"

- Egraphs
- (*SemanticLabel, StructuralElement*)

Semi-Supervised Approaches

AutoSlog TS [Riloff, 1996]

- Input: pre-classified documents (relevant vs. irrelevant)
- NLP as preprocessing: full parser for detecting subject-v-object relationships
- Principle
  - Relevant patterns are patterns occuring more often in the relevant documents
- Output: ranked patterns, but not classified, namely, only the left hand side of a template filling rule
- The dictionary construction process consists of two stages:
  - pattern generation and
  - statistical filtering
- Manual review of the results



AutoSlog-TS flowchart

Pattern Extraction

The sentence analyzer produces a syntactic analysis for each sentence and identified noun phrases. For each noun phrase, the heuristic rules generate a pattern to extract noun phrase.

<subject> bombed

Relevance Filtering

- the whole text corpus will be processed a second time using the extracted patterns obtained by stage 1.
- Then each pattern will be assigned with a relevance rate based on its occurring frequency in the relevant documents relatively to its occurrence in the total corpus.
- A preferred pattern is the one which occurs more often in the relevant documents.

Statistical Filtering

Relevance Rate:

$$Pr(\text{relevant text} \setminus \text{text contains case frame}_i) = \frac{rel - freq_i}{total - freq_i}$$

$rel\text{-}freq_i$ number of instances of $case - frame_i$ *in the relevant documents*

$total\text{-}freq_i$ total number of instances of $case - frame_i$

Ranking Function:

$$score_i = relevance\ rate_i * \log_2 (frequency_i)$$

$Pr < 0,5$ negatively correlated with the domain

„Top“

1. <subj> exploded
2. murder of <np>
3. assassination of <np>
4. <subj> was killed
5. <subj> was kidnapped
6. attack on <np>
7. <subj> was injured
8. exploded in <np>
9. death of <np>
10. <subj> took_place
11. caused <dobj>
12. claimed <dobj>
13. <subj> was wounded
14. <subj> occurred
15. <subj> was located
16. took_place on <np>
17. responsibility for <np>
18. occurred on <np>
19. was wounded in <np>
20. destroyed <dobj>
21. <subj> was murdered
22. one of <np>
23. <subj> kidnapped
24. exploded on <np>
25. <subj> died

The Top 25 Extraction Patterns

Empirical Results

- 1500 MUC-4 texts
  - 50% are relevant.
- In stage 1, 32,345 unique extraction patterns.
- A user reviewed the top 1970 patterns in about 85 minutes and kept the best 210 patterns.
- Evaluation
  - AutoSlog and AutoSlog-TS systems return comparable performance.

Conclusion

- Advantages
  - Pioneer approach to automatic learning of extraction patterns
  - Reduce the manual annotation
- Disadvantages
  - Ranking function is too dependent on the occurrence of a pattern, relevant patterns with low frequency can not float to the top
  - Only patterns, not classification

Unsupervised

ExDisco (Yangarber 2001)

- Seed
- Bootstrapping
- Duality/Density Principle for validation of each iteration

Input

- a corpus of unclassified and unannotated documents
- a seed of patterns, e.g.,

subject(company)-verb(appoint)-object(person)

NLP as Preprocessing

- full parser for detecting subject-v-object relationships
  - NE recognition
  - FDG formalism (Tapannaien & Järvinen, 1997)

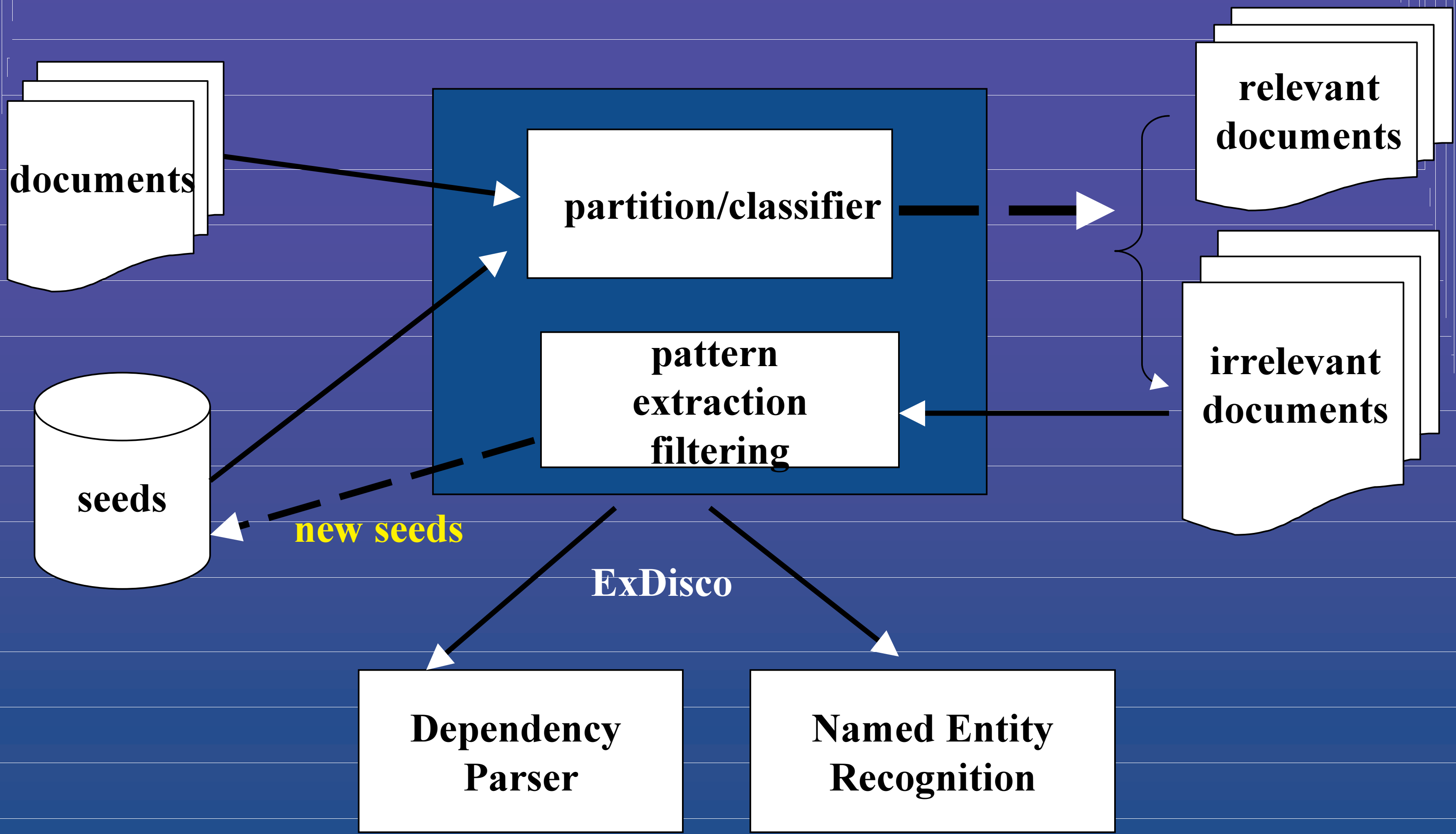
Duality/Density Principle (boostrapping)

- Density:
  - Relevant documents contain more relevant patterns
- Duality:
  - documents that are relevant to the scenario are strong indicators of good patterns
  - good patterns are indicators of relevant documents

Algorithm

- Given:
  - a large corpus of un-annotated and un-classified documents
  - a trusted set of scenario patterns, initially chosen ad hoc by the user, the seed. Normally is the seed relatively small, two or three
  - (possibly empty) set of concept classes
- Partition
  - applying seed to the documents and divide them into relevant and irrelevant documents
- Search for new candidate patterns:
  - automatic convert each sentence into a set of candidate patterns.
  - choose those patterns which are strongly distributed in the relevant documents
  - Find new concepts
- User feedback
- Repeat

Workflow



documents
partition/classifier
pattern
extraction
filtering
relevant
documents
irrelevant
documents
seeds
new seeds
ExDisco
Dependency
Parser
Named Entity
Recognition

Pattern Ranking

$$\text{Score}(P) = \frac{|H \cap R|}{|H|} \cdot \textbf{LOG}\,(|H \cap R|)$$

Evaluation of Event Extraction

Pattern Base	*Recall*	*Precision*	*F*
Seed	27	74	39.58
ExDisco	52	72	60.16
Union	57	73	63.56
Manual-MUC	47	70	56.40
Manual-NOW	56	75	64.04

ExDisco

- Advantages
  - Unsupervised
  - Multi-slot template filler rules
- Disadvantages
  - Only subject-verb-object patterns, local patterns are ignored
  - No generalization of pattern rules (see inductive learning)
  - Collocations are not taken into account, e.g., *PN take responsibility of Company*
- Evaluation methods
  - Event extraction: integration of patterns into IE system and test recall and precision
  - Qualitative observation: manual evaluation
  - Document filtering: using ExDisco as document classifier and document retrieval system

Relational learning and Inductive Logic Programming (ILP)

- Allow induction over structured examples that can include first-order logical representations and unbounded data structures

Semi-Structured and Un-Structured Documents

RAPIER [Califf, 1998]

- Uses relational learning to construct unbounded pattern-match rules, given a database of texts and filled templates
- Primarily consists of a bottom-up search
- Employs limited syntactic and semantic information
- Learn rules for the complete IE task

Filled template of RAPIER

Posting from Newsgroup

```
Telecommunications. SOLARIS Systems
Administrator. 38-44K. Immediate need

Leading telecommunications firm in need
of an energetic individual to fill the
following position in the Atlanta
office:

  SOLARIS SYSTEMS ADMINISTRATOR
  Salary: 38-44K with full benefits
  Location: Atlanta Georgia, no
        relocation assistance provided
```

Filled Template

```
computer_science_job
title: SOLARIS Systems Administrator
salary: 38-44K
state: Georgia
city: Atlanta
platform: SOLARIS
area: telecommunications
```

Figure 1: Sample Message and Filled Template

RAPIER's rule representation

- Indexed by template name and slot name
- Consists of three parts:

1. A pre-filler pattern
2. Filler pattern (matches the actual slot)
3. Post-filler

Pattern

- Pattern item: matches exactly one word
- Pattern list: has a maximum length N and matches 0..N words.
- Must satisfy a set of constraints
  1. Specific word, POS, Semantic class
  2. Disjunctive lists

RAPIER Rule

ORIGINAL DOCUMENT:	EXTRACTED DATA:
AI. C Programmer. 38-44K. Leading AI firm in need of an energetic individual to fill the following position:	computer-science-job title: C Programmer salary: 38-44K area: AI

AREA extraction pattern:

Pre-filler pattern:	word: leading
Filler pattern:	list: len: 2 tags: [nn, nns]
Post-filler pattern:	word: [firm, company]

RAPIER'S Learning Algorithm

- Begins with a most specific definition and compresses it by replacing with more general ones
- Attempts to compress the rules for each slot
- Preferring more specific rules

Implementation

- Least general generalization (LGG)
- Starts with rules containing only generalizations of the filler patterns
- Employs top-down beam search for pre and post fillers
- Rules are ordered using an information gain metric and weighted by the size of the rule (preferring smaller rules)

Example

Located in Atlanta, Georgia.
Offices in Kansas City, Missouri

Pre-filler:	Filler:	Post-filler:
1) word: located	1) word: atlanta	1) word: ,
tag: vbn	tag: nnp	tag: ,
2) word: in		2) word: georgia
tag: in		tag: nnp
		3) word: .
		tag: .

and

Pre-filler:	Filler:	Post-filler:
1) word: offices	1) word: kansas	1) word: ,
tag: nns	tag: nnp	tag: ,
2) word: in	2) word: city	2) word: missouri
tag: in	tag: nnp	tag: nnp
		3) word: .
		tag: .

Example (cont)

Pre-filler:	Filler:	Post-filler:
	1) list: max length: 2 word: {atlanta, kansas, city} tag: nnp	

and

Pre-filler:	Filler:	Post-filler:
	1) list: max length: 2 tag: nnp	

Pre-filler:	Filler:	Post-filler:
1) word: in tag: in	1) list: max length: 2 word: {atlanta, kansas, city} tag: nnp	1) word: , tag: ,

and

Pre-filler:	Filler:	Post-filler:
1) word: in tag: in	1) list: max length: 2 tag: nnp	1) word: , tag: ,

Example (cont)

Final best rule:

Pre-filler:	Filler:	Post-filler:
1) word: in	1) list: max length: 2	1) word: ,
tag: in	tag: nnp	tag: ,
		2) tag: nnp
		semantic: state

Experimental Evaluation

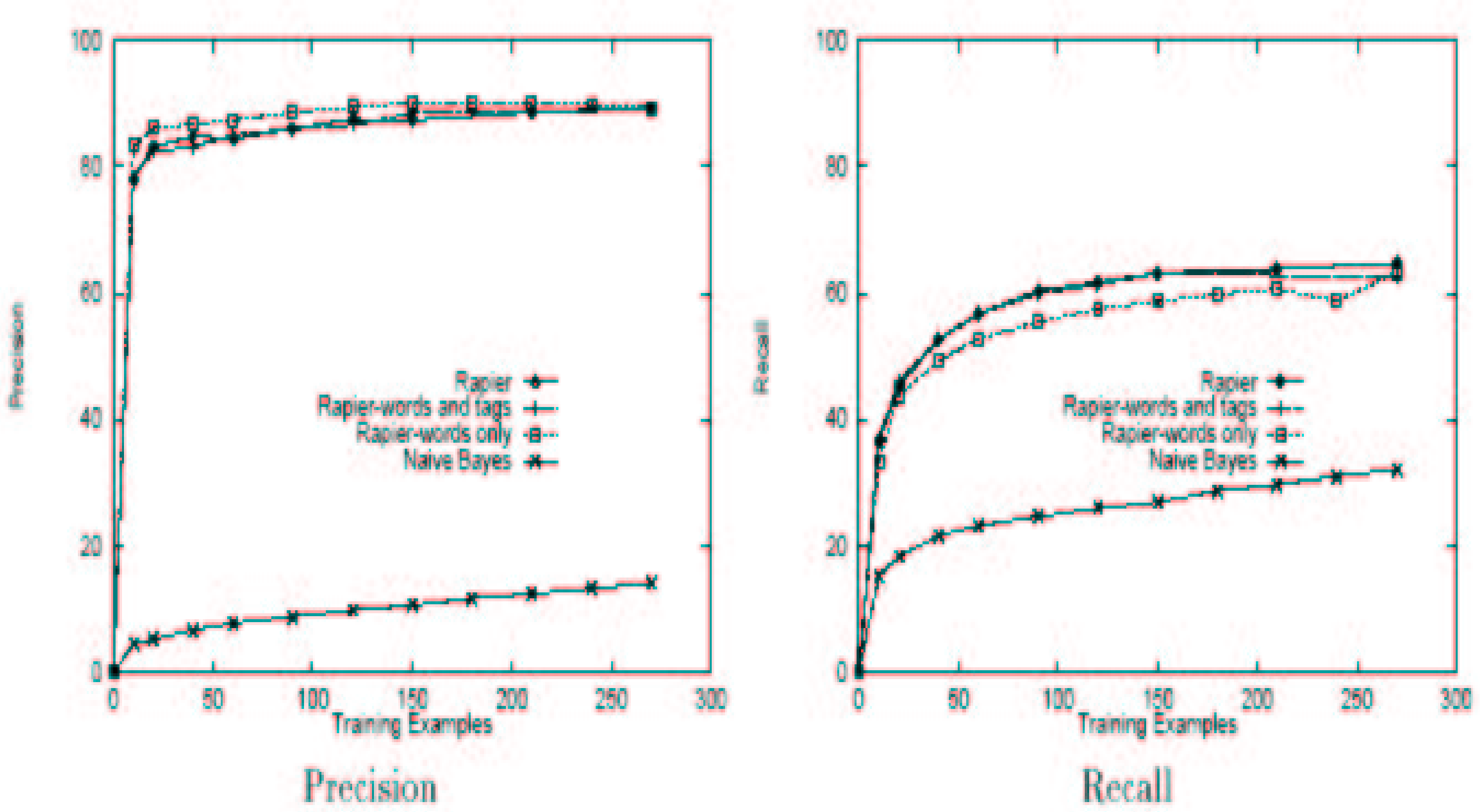
- A set of 300 computer-related job posting from austin.jobs
- A set of 485 seminar announcements from CMU.
- Three different versions of RAPIER were tested

1.words, POS tags, semantic classes

2. words, POS tags

3. words

Performance on job postings



Results for seminar announcement task

System	stime		etime		loc		speaker	
	Prec	Rec	Prec	Rec	Prec	Rec	Prec	Rec
RAPIER	93.9	92.9	95.8	94.6	91.0	60.5	80.9	39.4
RAP-WT	96.5	95.3	94.9	94.4	91.0	61.5	79.0	40.0
RAP-W	96.5	95.9	96.8	96.6	90.0	54.8	76.9	29.1
NAIBAY	98.2	98.2	49.5	95.7	57.3	58.8	34.5	25.6
SRV	98.6	98.4	67.3	92.6	74.5	70.1	54.4	58.4
WHISK	86.2	100.0	85.0	87.2	83.6	55.4	52.6	11.1
WH-PR	96.2	100.0	89.5	87.2	93.8	36.1	0.0	0.0

Conclusion

- Pros
  - Have the potential to help automate the development process of IE systems.
  - Work well in locating specific data in newsgroup messages
  - Identify potential slot fillers and their surrounding context with limited syntactic and semantic information
  - Learn rules from relatively small sets of examples in some specific domain
- Cons
  - single slot
  - regular expression
  - Unknown performances for more complicated situations