A Comparison of Statistical and Rule-Induction Learners for Automatic Tagging of Time Expressions in English

14th International Symposium on Temporal Representation and Reasoning

(TIME'07)

Jordi Poveda, Mihai Surdeanu and Jordi Turmo
TALP Research Center
Technical University of Catalonia (UPC)
Barcelona, Spain
{jpoveda,surdeanu,turmo}@lsi.upc.edu

June 28st, 2007

1. Time Expression Recognition
   - Information Extraction
   - TERN (Time Expression Recognition and Normalization)
2. Machine Learning for TE Recognition
   - Problem description: Chunking
   - Statistical: Support Vector Machines
   - Rule Induction: Inductive Logic Programming
3. Results
   - Experiments
   - Support Vector Machines
   - Inductive Logic Programming
   - Comparison
4. Conclusions

Example

"*Yesterday*, *German giant E.ON*'s board of directors announced plans for *takeover* of *Spanish ENDESA* for *$20 million* at *an undisclosed date*, just *after* receiving *former CEO Bernotat*'s *resignation* notice." from Reuters, 11-24-2005

1. 1 takeover EVENT:

   takeover_EVENT(id(EVENT1), acquirer(E.ON), target(ENDESA), amount($20 million), date(TIME1))

2. 1 resignation EVENT:

   resignation_EVENT(id(EVENT2), company(E.ON), person(Bernotat), position(CEO))

3. 1 precedes RELATION:

   precedes(EVENT2, EVENT1)

4. 2 time expressions:

   timex(id(TIME1), type(date), mention(an undisclosed date), value(??-??-????))

   timex(id(TIME2), type(date), mention(yesterday), value(11-23-2005))

Definition

Information Extraction (IE) is a subtask in Natural Language Processing whose objective is *extracting information from unstructured machine-readable documents and arranging it into an structured, processable form*.

Information is usually represented in a *relational form*, or structured by using *metadata* such as *XML tags*.

Objectives:

- Populating relational databases
- Monitoring information sources within a domain (e.g. news feeds on corporate mergers and acquisitions)
- Inference
- Structuring information for use in other NLP problems: QA (Question Answering), AS (Automatic Summarisation), . . .

Example

To *identify* (Recognition) the mentions in text of time-denoting expressions and to *capture their meaning* in a canonical form (Normalization)

```
But even <TIMEX2 VAL="1999-07-22"> last Thursday
</TIMEX2>, there were signs of potential battles
<TIMEX2 VAL="FUTURE_REF" ANCHOR_DIR="AFTER"
ANCHOR_VAL="1999-07-22"> ahead </TIMEX2>.
```

L. Ferro, L. Gerber, I. Mani, B. Sundheim, and G. Wilson. *TIDES Standard for the Annotation of Temporal Expressions v1.3*. Technical Report, MITRE Corporation, 2003.

Examples of time expressions

- Fully-specified time references:

  16th June 2006, *the twentieth century*, *Monday at 3pm*
- Context-dependent:

  the previous month, *three days after the meeting*, *February the following year*
- Anaphoric and relative to the time when the expression is written:

  that day, *yesterday*, *currently*, *then*
- Durations or intervals:

  a month, *three days*, *some hours in the afternoon*
- Frequencies or recurring times:

  monthly, *every other day, once a week*, *every first Sunday of a month*
- Culturally dependent time denominations:

  Easter, *the month of Ramadan*, *St. Valentine*
- Fuzzy or vaguely specified time references:

  the future, *some day*, *eventually*, *anytime you so desire*

1. Time Expression Recognition
   - Information Extraction
   - TERN (Time Expression Recognition and Normalization)
2. Machine Learning for TE Recognition
   - Problem description: Chunking
   - Statistical: Support Vector Machines
   - Rule Induction: Inductive Logic Programming
3. Results
   - Experiments
   - Support Vector Machines
   - Inductive Logic Programming
   - Comparison
4. Conclusions

Chunking

Chunking: Assigning B (begin), I (inside), O (outside) tags to each token in a sequence

`But/O even/O last/B Thursday/I ,/O there/O were/O signs/O of/O potential/O battles/O ahead/B ./O`

- Limited to non-overlapping, non-recursive chunks (i.e. a chunk inside a longer chunk)
- Chunk need not be bounded in length

Token features (I)

- Lexical:
  Token form, token in lowercase, token w/o alphabetic chars (e.g. 3 for "3pm"), the token w/o alphanumeric chars (e.g. - - - for 1995-07-12)
- Morphological: POS (Part Of Speech) tag
  (e.g. NN $\rightarrow$ noun, JJ $\rightarrow$ adjective, CD $\rightarrow$ cardinal number, MD $\rightarrow$ modal verb)
- Syntactic: Basic syntactic chunk type
  (e.g. I-NP $\rightarrow$ inside noun phrase, B-VP $\rightarrow$ beginning of verb phrase, ...)
- Format features:
  1. *isAllCaps* like "THU"
  2. *isAllCapsOrDots* like "I.B.M"
  3. *isAllDigits* like "2004"
  4. *isAllDigitsOrDots* like "10.24"
  5. *initialCap* like "February"

Token features (II)

- Bag-of-words:
  1. *isNumber* (e.g. *one*, *two*, *ten*, . . .)
  2. *isMultiplier* (e.g. *hundred*, *thousands*, . . .)
  3. *isDay* (e.g. *monday*, *mon*, *saturday*, *sat*, . . .)
  4. *isMonth* (e.g. *january*, *jan*, *june*, *jun.*, . . .)
- Contextual features: All of the above features, w.r.t. context tokens
- Dynamic features: The BIO tags for a window of previous tokens

YamCha (Yet Another Multi-purpose CHunk Annotator)

- YamCha [1]: Multipurpose chunker based on SVM (Vapnik, 1995)
- SVMs: Max-margin discriminative classifiers based on quadratic optimization
- Map a feature vector into a vector space of higher dimension, exploring combinations of features ("kernel trick")
- Requires with numeric features: 1 categorial feature with N tags $\rightarrow$ N binary features
- One-vs-rest classification: Train 3 classifiers (B against I/O, I against B/O, O against B/I)
- Classifiers' outputs are combined based on margins and previous tokens

[1] http://chasen.org/~taku/software/yamcha/

Sample training data

	FORM	POS tag	SYNTAX	BIO tag
POS: -3	But	CC	O	O
POS: -2	even	RB	B-ADVP	O
POS: -1	last	JJ	B-NP	B-TIMEX
POS: 0	Thursday	NNP	I-NP	I-TIMEX
POS: +1	,	,	O	O
POS: +2	there	EX	B-NP	O
POS: +3	were	VBD	B-VP	O

Training: FOIL

- Inductive Logic Programming (ILP) attempts to learn a *logic program* for a set of target concepts from:
  - Target *predicates*: $p_i(X_1, \ldots, X_{n_i})$
  - Examples and counterexamples $\mathcal{E}$: ground facts $< x_1, \ldots, x_n >$
  - Background knowledge predicates $\mathcal{B}$: $q_i(X_1, \ldots, X_{m_i})$
  - Hypothesis language $\mathcal{L}$
- FOIL: An empirical (top-down, non-interactive) ILP system (Quinlan, 1993)
- The hypothesis language of FOIL are Horn clauses without functions
- Train 3 objective predicates: one for B (begin), one for I (inside), one for O (outside)
- Background knowledge predicates are the token features described earlier

Sample input and output

Input:

```
form_last(tok100).  // token 100 is 'last'
form_Thursday(tok101).  // token 101 is 'Thursday'
POS_NNP(tok101).  // token 101 is a proper noun
syn_I_NP(tok101).  // token 101 is inside a noun phrase
context_r1_form_Thursday(tok100).  // token right of tok100 is 'Thursday'
context_l1_B_NP(tok101).  // token left of tok101 is at the start of a
noun phrase
                                   ...
```

Output:

```
begin_timex(X) :- form_Thursday(X).
begin_timex(X) :- syn_I_NP(X), context_l1_B_PP(X),
not(context_l1_form(with)).
                                   ...
inside_timex(X) :- form_ago(X), context_l2_POS_CD(X).
inside_timex(X) :- POS_CD(X), not(context_l1_t_O(X)).
                                   ...
```

Evaluation: PROLOG

- For evaluation, load learned predicates into PROLOG and a knowledge base with declarations of all the token features in the test data
- More than one predicate B/I/O can return *yes* for a given token → Combination of classifiers' outputs
- Assign a *confidence* to each learned rule (supporting evidence): $conf(A \Leftarrow B) = \frac{\#(A \wedge B)}{\#B}$
- Two approaches:
  1. "best" → Take $conf(A \Leftarrow B)$ to be that of best clause satifistied by token
  2. "sum" → Take $conf(A \Leftarrow B)$ to be the sum of confidences of all satisfied clauses
- Enforce consistency rule: I cannot follow O or be the first tag beginning a sentence
- If all three B/I/O return *no* → assign most probable (O)

Corpus

- ACE (Automatic Content Extraction) 2005 corpus
- 550 documents from five categories (NW, BN, BC, CTS and WL)
- 257K tokens, 8809 tokens in time expressions (3.42%), 4650 time expression mentions
- 80% for training, 20% for testing

Experiments

- Support Vector Machines (YamCha):
  - temp. cost for training $= 8 \pm 4$ hours

1. 5-fold cross-validation with optimal parameters
2. Incremental feature sets
3. Varying kernel degree $(1 \ldots 3)$
4. Varying context window size $(1 \ldots 3)$

- ILP (FOIL):
  - temp. cost for training = in the order of weeks

1. Same optimal parameters as SVM
2. Simplifying the training data

Measures

Precision: The rate of returned temporal expressions that are correctly identified (i.e. correctly tagged divided by total tagged).

Recall: The rate of existing temporal expressions that are correctly identified (i.e. correctly tagged divided by those that should have been tagged).

F_1 Score: It is the harmonic mean of the two previous values: $F_1 = \frac{2 \times Precision \times Recall}{(Precision + Recall)}$.

Accuracy: The percentage of correct BIO tag assignments predicted by the classifier at the token level (i.e. whether the predicted tag coincides with the target tag).

Optimal Model

	PREC	RECALL	F_1	ACC
Round 1	81.33	75.23	78.16	98.68
Round 2	77.74	70.46	73.92	98.60
Round 3	75.92	71.22	73.50	98.47
Round 4	80.05	73.71	76.75	98.65
Round 5	80.34	72.54	76.24	98.66
AVERAGE	**79.08**	**72.63**	**75.71**	**98.61**
STD DEV.	2.20	1.91	1.97	0.17

Degree of polynomial kernel

KERNEL	PREC	RECALL	F_1	ACC
pol. lineal	72.39 (-7.66)	70.08 (-3.63)	71.21 (-5.54)	98.25 (-0.40)
pol. quadratic	**80.05**	**73.71**	**76.75**	**98.65**
pol. cubic	81.30 (+1.25)	71.73 (-1.98)	76.21 (-0.54)	98.65 (+0.00)

Incremental Feature Sets

FEATURES	PREC	RECALL	F_1	ACC
Model 1	80.00 (-0.05)	66.89 (-6.82)	72.86 (-3.89)	98.56 (-0.09)
Model 2	80.10 (+0.05)	71.73 (-1.98)	75.68 (-1.07)	98.60 (-0.05)
Model 3	**80.05**	**73.71**	**76.75**	**98.65**

- Model 1: token form + lowercase
- Model 2: Model 1 + POS tags + format features (isAllCaps, isAllDigits, etc) + form w/o alphabetic chars + form w/o alphanumeric chars
- Model 3: Model 2 + syntactic chunks + bag-of-words (isNumber, isMultiplier, isDay, isMonth)

Context window size

WINDOW	PREC	RECALL	F_1	ACC
-1 .. +1	74.47 (-5.58)	72.83 (-0.88)	73.64 (-3.11)	98.41 (-0.24)
-2 .. +2	**80.05**	**73.71**	**76.75**	**98.65**
-3 .. +3	80.30 (+0.25)	71.29 (-2.42)	75.52 (-1.23)	98.59 (-0.06)

Optimal model (for SVM)

CLASSIFIER	PREC	RECALL	F_1	ACC
FOIL (best)	77.58	**52.15**	**62.37**	97.95
FOIL (sum)	**81.32**	50.28	62.13	**97.98**

best $\longrightarrow$ Take $conf(A \Leftarrow B)$ to be that of best clause satifistied by token
sum $\longrightarrow$ Take $conf(A \Leftarrow B)$ to be the sum of confidences of all satisfied clauses

Reducing model complexity

- Unaffordable temporal complexity with the full model (over $3\frac{1}{2}$ weeks each classifier B/I/O)
- With 1-arity predicates, FOIL's complexity is quadratic on $\|\mathcal{B}\|$ (predicates) and $\|\mathcal{E}\|$ (examples)
- Reducing the volume of the training data:
  1. Filtering less common predicates
  2. Filtering less relevant counterexamples
- Temporal cost considerably reduced (in the order of days), at the expense of approx. -8% prec/recall

SVM and ILP side by side

CLASSIFIER	PREC	RECALL	F_1	ACC
FOIL	81.32 (+1.27)	52.15 (-21.56)	62.37 (-14.38)	97.98 (-0.67)
SVM	**80.05**	**73.71**	**76.75**	**98.65**

SVM clearly superior

Final thoughts

- ILP is a dead end: elegant representation for "toy" problems and/or small datasets, unusable for large corpora
- Alternatives approaches for rule induction: Statistical Rule Learning, simpler rule languages (propositional, N-term clauses), semi-supervised IE pattern learning
- Combination methods (Statistical + Rules)
- Machine-learning vs. grammar-based approaches (complementary?)
  - Best performance with statistical ML around 80% (depending on "feature engineering" and training corpus size)
  - Best performance with handwritten grammars around 90%-95%
  - Difficult to define a grammar to cover difficult cases ("easy" cases account for a majority)
  - Grammars must be specifically written for each new extraction domain
  - On the other hand, Normalization lends itself to the grammar approach

Thanks

Many thanks for your attention
Any questions? Comments?