Named Entity Recognition --machine learning methods

Contents

- **Models Introduction**

  CRF model

- **NE identification using Machine Learning Approach**

① **Supervised learning**

② **Semi-supervised learning**

Machine Learning method (idea)

- Based on Probability:

“smith was appointed as CEO of IBM”

Category: *Person*,*position*,*company*,nn.

If we know:

P(smith | person)=0.8 √

P(smith | company)=0.1

P(smith | position) =0.05

P(smith | nn) =0.05

Machine learning method (idea)

- Sequence labeling:

W： smith was appointed as CEO of IBM

NC :Per nn nn nn Pos nn CO

NC: CO Per nn nn Per nn Per

……

P(NC sequences | W sequence)

Which NC sequences has a larger probability?

Machine learning method (idea)

- Classification

W： smith was appointed as CEO of IBM

With sliding window

Classification Model

Category: *Person*,*position*,*company*,nn

Machine learning for NE recognition

- **Supervised Learning**
  - Training is based on available very large annotated corpus.
  - Mainly statistical-based models used

1. *Bigram model*
2. *HMM (Hidden Markov Model)*
3. *CRF (conditional random field)*

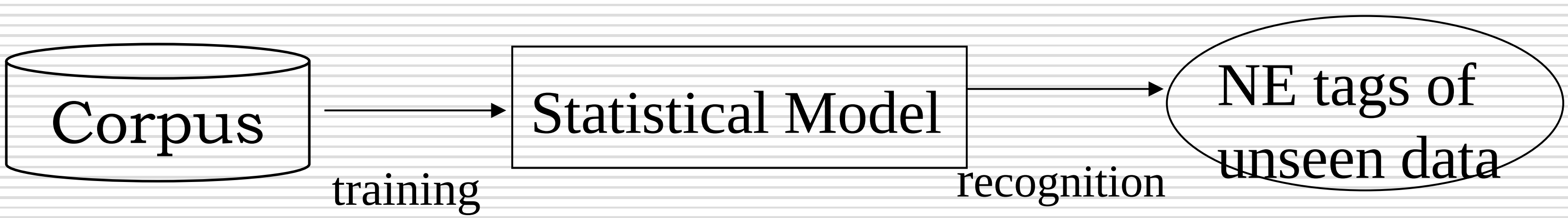
Supervised Machine Learning for NE Recognition

1. ***Construct a training corpus by manual annotation***

   e.g. I lived in <LO>Beijing</LO>

   Extract necessary statistics from the corpus to build a statistical model which can automatically estimate Pr(NC Sequence | W Sequence) for unseen data.

2. ***For any unseen data**, based on the statistical model to search the NC sequence which maximizes the probability Pr(NC Sequence | W Sequence).*



Statistical Model for Named Entity Recognition

- *Given a sequence of words (W), the goal of NE recognition is to find the sequence of name-class (NC) with maximum Pr(NC|W).*

$$\operatorname{argmax}_{\text{nc sequence}} \Pr(\text{NCSequence} \mid \text{W Sequence})$$

e.g, *given word sequence :*

it has set up a joint venture in Hong Kong

possible name-class sequence (LO: location OR: organization)

it	has	set	up	a	joint	venture	in	Hong	Kong
NN	NN	NN	NN	NN	NN	NN	NN	LO	LO
LO	NN	NN	NN	NN	NN	NN	NN	OR	LO

Encoding classes for sequence labeling

- **IO** encoding

Fred showed Sue Mengqiu Huang

Per nn Per Per Per

- **IOB** encoding

Fred showed Sue Mengqiu Huang

B-Per nn B-Per B-Per I-Per

IO encoding is simple, much fast than IOB encoding

N-gram Model for NE Recognition

- **Question:** *How to evaluate Pr(NC Sequence| Sentence) based on unigram and bigram information?*
- ***One solution***: *transfer the conditional probability into (NC,Sentence) joint probability (Bayes' equation)*
- *Decouple a sentence into bigram sequences (Markov assumption)*

Bayes Equation

Based on Bayes equation:

$$\mathrm{argmax}_{nc\ sequence} \Pr(NC\,Sequence \mid W\,Sequence)$$

$$= \mathrm{argmax}_{nc\ sequence} \frac{\Pr(W\,Sequence, NC\,Sequence)}{\Pr(W\,Sequence)}$$

$$= \mathrm{argmax}_{nc\ sequence} \Pr(W\,Sequence, NC\,Sequence,)$$

Markov Assumption

$$\Pr(\text{NCSequence}, \text{W Sequence})$$

$$= \Pr(w_n, nc_n, w_{n-1}, nc_{n-1}, \ldots, w_0, nc_0)$$

$$= \Pr(w_0, nc_0)\Pr(w_1, nc_1 \mid w_0, nc_0)\Pr(w_2, nc_2 \mid w_1, nc_1, w_0, nc_0)$$

$$\ldots\Pr(w_n, nc_n \mid w_{n-1}, nc_{n-1}, \ldots, w_0, nc_0)$$

Bigram Markov assumption:

$$\Pr(w_2, nc_2 \mid w_1, nc_1, w_0, nc_0) = \Pr(w_2, nc_2 \mid w_1, nc_1)$$

……………………………………

$$\Pr(w_n, nc_n \mid w_{n-1}, nc_{n-1}, \ldots, w_0, nc_0) = \Pr(w_n, nc_n \mid w_{n-1}, nc_{n-1})$$

Bigram-based NE Tagger

So the final formula is:

$$\Pr(NCSequence, WSequence)$$
$$= \Pr(w_0, nc_0)\Pr(w_1, nc_1 \mid w_0, nc_0)\Pr(w_2, nc_2 \mid w_1, nc_1)$$
$$\ldots\Pr(w_n, nc_n \mid w_{n-1}, nc_{n-1})$$

The size of the training corpus is large enough to provide fairly good bigram *information.*

Parameter Estimation based on Bayesian Analysis

- Question: how to estimate model parameters, *i.e.*

$$\Pr(w_n, nc_n \mid w_{n-1}, nc_{n-1})$$

- Parameter estimation based on Bayesian analysis: select parameters which maximize

$$\Pr(\text{parameter} \mid \text{training corpus})$$

- Based on Bayes equation, this is equivalent to maximize

$$\Pr\left(\underset{\uparrow}{\text{parameter}}\right)\Pr\left(\text{training corpus} \mid \text{parameter}\right)$$

Prior Probability

Maximum Likelihood Estimation

- The aim of maximum likelihood estimation (MLE) is to find the parameter value(s) that can predict the training corpus with the highest probability.

$$\text{argmax}_{\text{parameter}} \Pr(\text{training corpus} \mid \text{parameter})$$

i.e. the prior probability is neglected. C is the count

- The MLE for the bigram statistical NE tagger:

$$\Pr(w_n, nc_n \mid w_{n-1}, nc_{n-1}) = \frac{C(w_n, nc_n, w_{n-1}, nc_{n-1})}{C(w_{n-1}, nc_{n-1})}$$

Smoothing (平滑技术)

- limited size of training corpus → MLE suffers from training data over-fitting. MLE simply assign zero or even $\frac{0}{0}$ probabilities to unseen events.

- Smoothing: *add one* or modify MLE by taking the sampling space into consideration, *e.g. backing-off to estimations with larger sampling space*

$$\Pr(w_n, nc_n \mid w_{n-1}, nc_{n-1}) = \frac{C(w_n, nc_n, w_{n-1}, nc_{n-1}) + 1}{C(w_{n-1}, nc_{n-1}) + 1}$$

Smoothing (平滑技术)

- Unseen bigrams.

e.g. Input sentence: *Patt Gibbs*

$\Pr(Gibbs, \mathrm{nc}_{Gibbs.} \mid Patt, \mathrm{nc}_{Patt}) = 0$

- Smoothing: modify MLE by taking the sampling space into consideration, *e.g. backing-off to estimations with larger sampling space*

$$\Pr\left(Gibbs, NC_{Gibbs} \middle| \mathrm{P}att, NC_{\mathrm{Patt}}\right)$$
$$\approx \lambda \Pr\left(Gibbs, NC_{Gibbs} \middle| NC_{\mathrm{Patt}}\right)$$

CRF model

- Lafferty, Pereira, and McCallum proposed this model in 2001
- A best model for named entity recognition
- A sequence model, the theory is complicated and omitted.
- Training is slow

General Working Flow

- Training

1. Collect representative training documents
2. Label each token for its entity or other (nn)
3. Design feature extractors (templates)
4. Train the sequence model

- Testing

1. Input test documents
2. Run sequence model to predict labels for each token
3. Correctly output the recognized entities (match the output format)

Features for sequence labeling

- Words
  - Current word (essentially like a learned dictionary)
  - Previous/next word (context)
- Other kinds of inferred linguistic classification
  - Part-of-speech tags
- Label context
  - Previous (and perhaps next) label

For example

Feature Notation	**Comment**
w_0	Current word (token)
w_1	Next word
w_{-1}	Previous word
w_0w_1	Current and next
$w_{-1}w_0$	Previous and current
$w_{-1}w_1$	Previous and next
w_2	Next next
…	… and friends

Features commonly used in training named entity recognition systems

- identity of wi
- identity of neighboring words
- part of speech of wi
- part of speech of neighboing words
- base-phrase syntactic chunk label of wi and neighboring words
- presence of wi in a gazeteer
- wi contains a particular prefix (from all prefixes of length<=4)
- wi contains a particular suffix (from all suffixes of length<=4)
- wi is all upper case
- word shape of wi
- word shape of neighboring words
- short word shape of wi
- short word shape of neighboring words
- presence of hyphen

Training corpus （for CRF）

Word	POS	Chunk	Short shape	Label
American	NNP	B-NP	Xx	B-ORG
Airlines	NNPS	I-NP	Xx	I-ORG
,	,	O	,	O
a	DT	B-NP	x	O
unit	NN	I-NP	x	O
of	IN	B-PP	x	O
AMR	NNP	B-NP	X	B-ORG
Corp.	NNP	I-NP	Xx.	I-ORG
,	,	O	,	O
immediately	RB	B-ADVP	x	O
matched	VBD	B-VP	x	O
the	DT	B-NP	x	O
move	NN	I-NP	x	O
,	,	O	,	O
spokesman	NN	B-NP	x	O
Tim	NNP	I-NP	Xx	B-PER
Wagner	NNP	I-NP	Xx	I-PER
said	VBD	B-VP	x	O
.	,	O	.	O

Feature Template

- Each line is a template, special macro %x[row,col] is used to specify a token in the input file.

```
Input: Data
He          PRP   B-NP
reckons     VBZ   B-VP
the         DT    B-NP << CURRENT TOKEN
current     JJ    I-NP
account     NN    I-NP
```

template	expanded feature
%x[0,0]	the
%x[0,1]	DT
%x[-1,0]	rokens
%x[-2,1]	PRP
%x[0,0]/%x[0,1]	the/DT
ABC%x[0,1]123	ABCDT123

Feature Template (cont.)

- When you give a template "U01:%x[0,1]", CRF++ automatically generates a set of feature functions (func1 ... funcN) like:

```
func1 = if (output = B-NP and feature="U01:DT") return 1 else return 0
func2 = if (output = I-NP and feature="U01:DT") return 1 else return 0
func3 = if (output = O and feature="U01:DT") return 1  else return 0
....
funcXX = if (output = B-NP and feature="U01:NN") return 1  else return 0
funcXY = if (output = O and feature="U01:NN") return 1  else return 0
...
```

Example of a template

```
# Unigram
U00:%x[-2,0]
U01:%x[-1,0]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[2,0]
U05:%x[-1,0]/%x[0,0]
U06:%x[0,0]/%x[1,0]

U10:%x[-2,1]
U11:%x[-1,1]
U12:%x[0,1]q
U13:%x[1,1]
U14:%x[2,1]
U15:%x[-2,1]/%x[-1,1]
U16:%x[-1,1]/%x[0,1]
U17:%x[0,1]/%x[1,1]
U18:%x[1,1]/%x[2,1]

U20:%x[-2,1]/%x[-1,1]/%x[0,1]
U21:%x[-1,1]/%x[0,1]/%x[1,1]
U22:%x[0,1]/%x[1,1]/%x[2,1]

# Bigram
B
```

- Unigram template: first character, 'U`
- Bigram template: first character, 'B`
- U01:identifiers for distinguishing relative positions.

Training

- Crf_learn *template_file train_file* **model_file**

- Parameters

-a CRF-L2 or CRF-L1: changing the regularization algorithm.

-c float: larger c, CRF tends to overfit to the given training corpus.

-f NUM: cut-off threshold. Use the features that occurs no less than NUM times in the given training data.

-p NUM: use multi-threading to faster the training step. NUM is the number of threads.

Testing

- Crf_test –m model test_file
- Parameter

-v sets verbose level. Default value is 0, Level 1 gives probabilities for each tag, and a conditional probability for the output.

-n best outputs: get n-best results sorted by the conditional probability of CRF

Software – CRF++

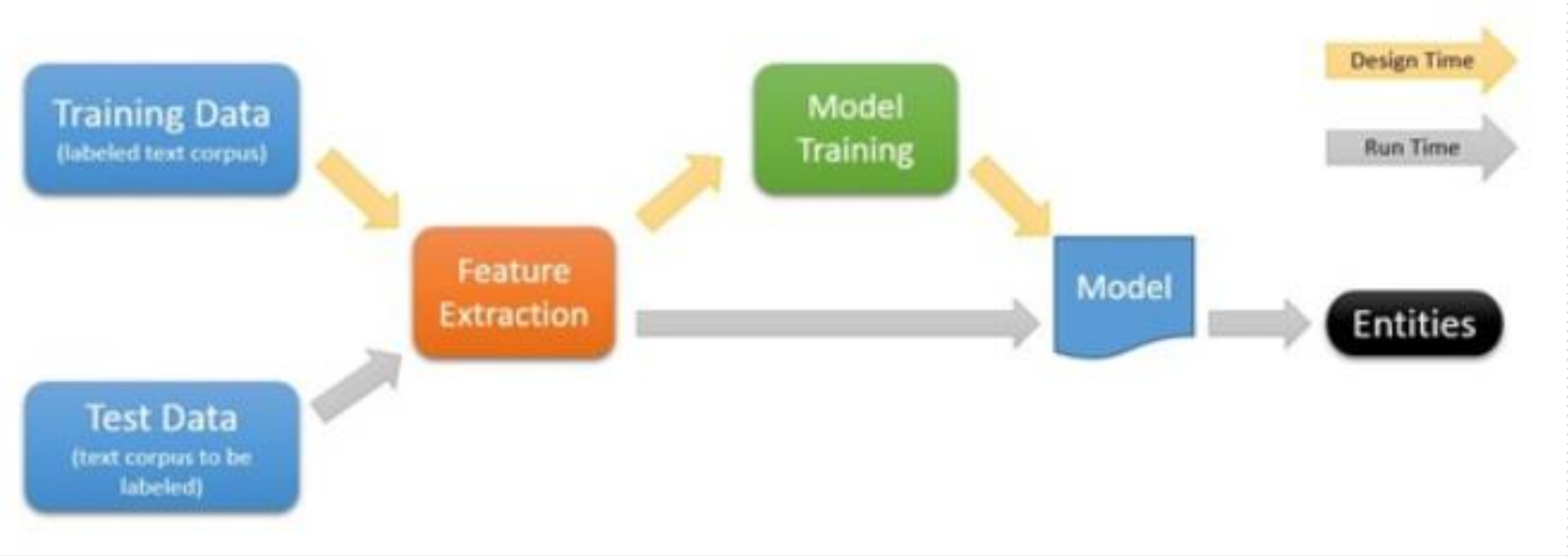
- http://code.google.com/p/crfpp/ , its homepage is now at: http://crfpp.googlecode.com/svn/trunk/doc/index.html ,
- Easy to use input & output format
- http://crfpp.sourceforge.net/

Supervised Machine Learning and Knowledge **Bottleneck**

- *Requires considerable size of training corpus, hence facing serious* knowledge bottleneck.
- *cannot effectively support user-defined named entities which are important for open-domain IE*

Supervised Machine learning Method (summarization)

- Feature extraction: very important.
- Model selection:



Chinese named entity results (CRF+MEM) in 2006

	Precision	Recall	F-score
LOC	94.19%	87.14%	90.53
ORG	83.59%	80.39%	81.96
PER	92.35%	74.66%	82.57

Table 2: The performance of the msra_a run broken down by entity type.

	Precision	Recall	F-score
LOC	93.09%	87.35%	90.13
ORG	75.51%	78.51	76.98
PER	91.52	79.27	84.95

Table 3: The performance of the msra_b run broken down by entity type.

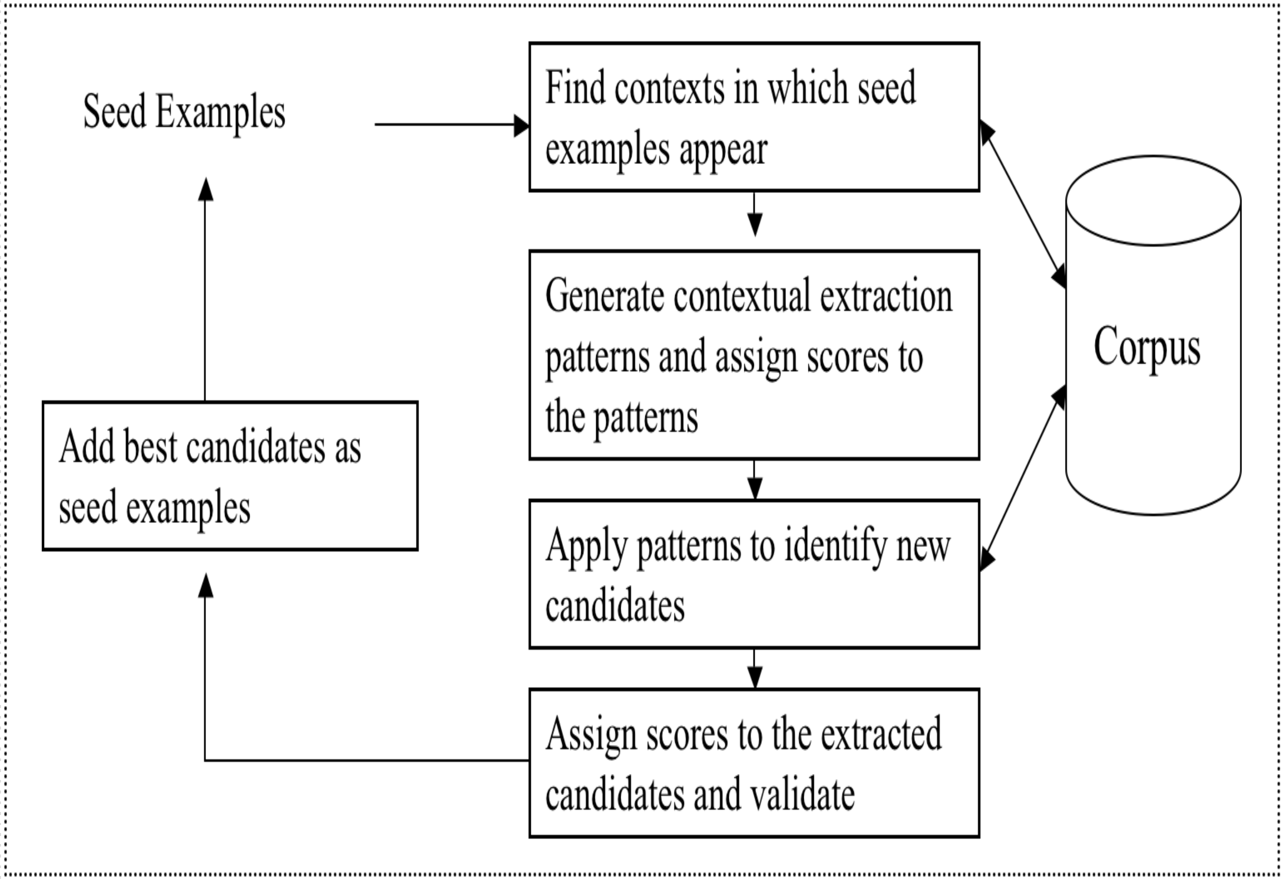
Semi-supervised method

- Training Corpus:
  - ✓ few seeds and large un-annotated corpus
- Training methods:
  - ✓ Bootstrapping
  - ✓ Others Expansion methods

Named Entity recognition based on semi-supervised learning (basic idea)

- *Define manually a small set of trusted seeds*
- *Training then only uses un-labeled data*
- *Initialize system by labeling the corpus with the seeds*
- *Extract and generalize patterns from the context of the seeds*
- *Use the patterns to further label the corpus and to extend the seed set (bootstrapping & expansion)*
- *Repeat the process unless no new terms can be identified.*

Bootstrapping Algorithm



Bootstrapping refers to a technology that starts from a small initial effort and gradually grows into something larger and more significant.

For Example

- Seeds: 腾讯公司
- Find contexts in which seeds appear
  - ✓ 腾讯公司CEO马化腾说。。。
  - ✓ 7月21日，腾讯宣布启动AI加速器
  - ✓ 腾讯宣布成立人工智能医学影像联合实验室
- Generate pattern based on the context
  - ✓ XX company CEO → XX is a company name
  - ✓ XX announced → XX is a company name
- Apply the pattern to find new one
  - 虹华公司CEO在一个大会上… → Hong Hua is a company
  - 华为宣布进军欧洲市场 → Huawei is a company
  - 某公司CEO撤销了。。。 →Some is not a company

How to score the patterns?

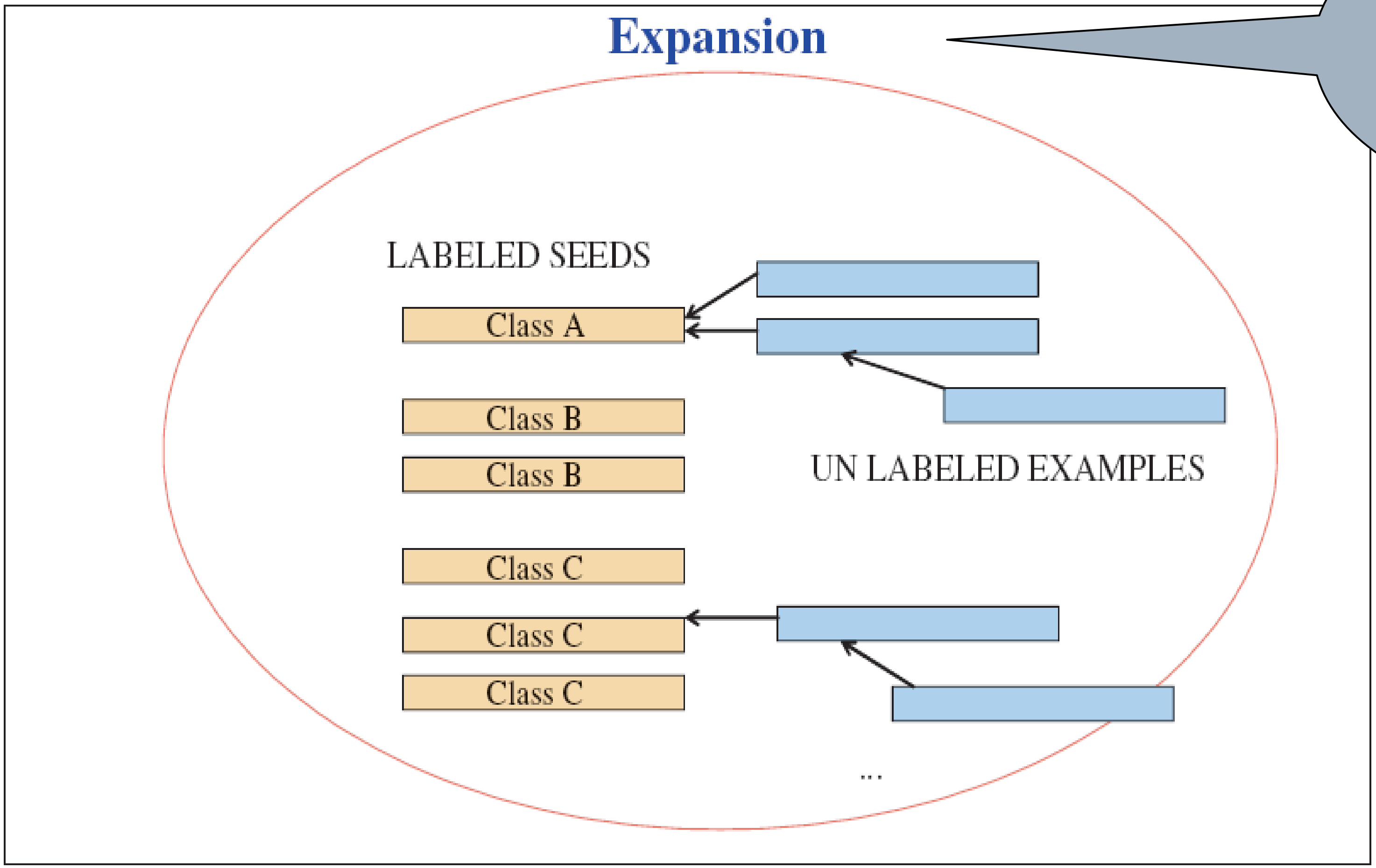
In order to find highly relevant or highly frequent **patterns**:

- relevance rate: $R_i = F_i / N_i$
  - F_i : the number of instances of pattern i that were activated in the positive examples.
  - N_i: the total number of instances of pattern i activated in the training corpus
- $score_i = R_i * \log(F_i)$

Bootstrapping algorithm (summarization)

- Starts with a small number of seed examples.
- Finds occurrences of these examples in a large set of documents.
- Generates contextual extraction patterns (rules) and assigns confidence scores to the patterns.
- Applies the extraction patterns to the documents and extracts new candidates.
- Assigns scores to the extracted candidates, and chooses the best ones to add to the seed set.
- Perform many similar iterations, and at every iteration it learns **more patterns** and can extract **more instances**.

Other Learning Methods

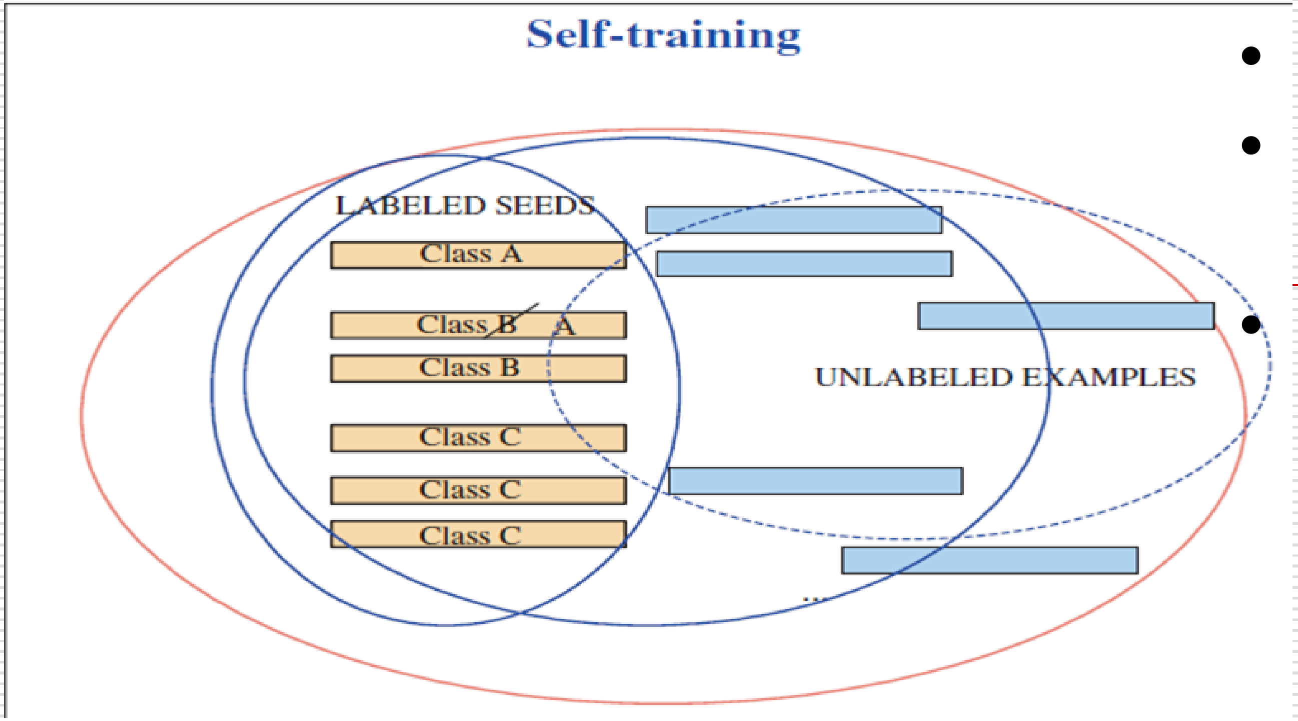


learn to classify unlabeled example to the closest seeds

How to identify unlabeled example based on seeds?

Many Techniques besides bootstrapping:

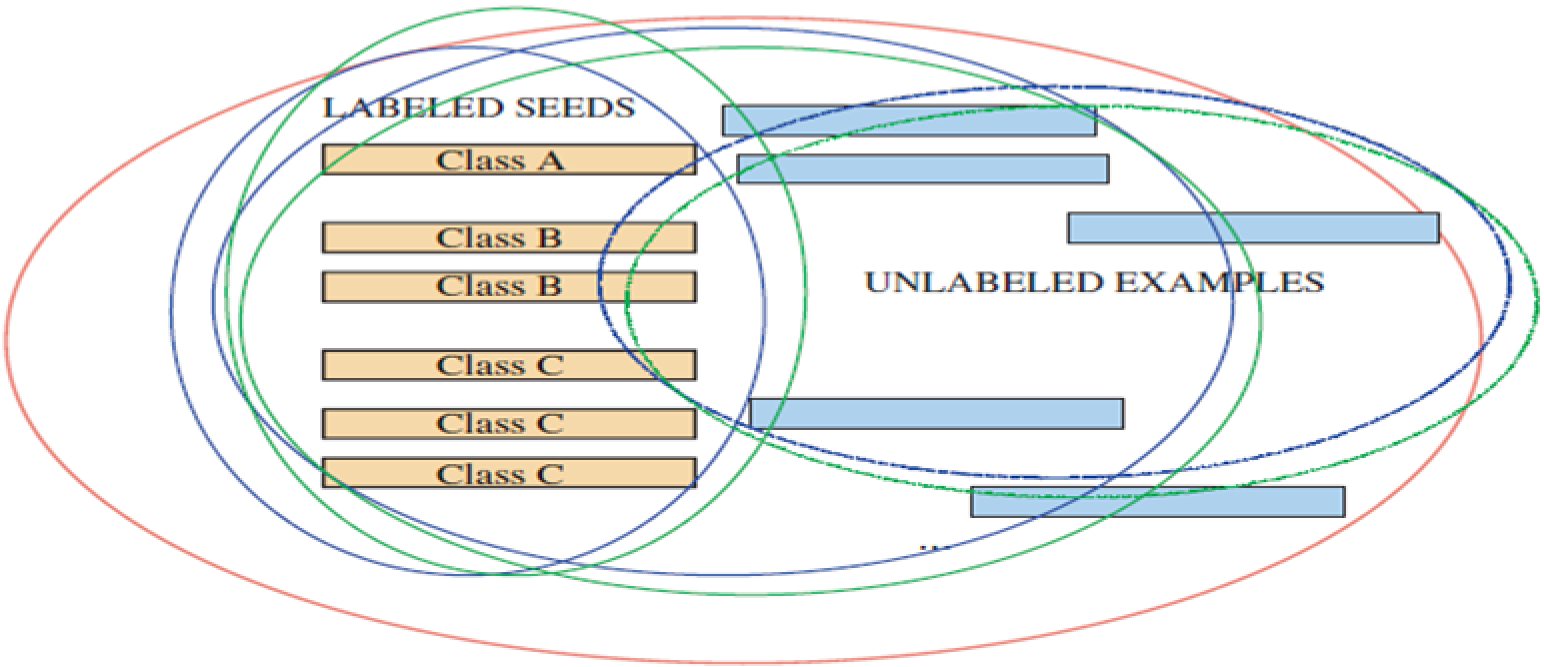
- Self-learning
- Co-training
- Active learning
- …



- Incrementally
- supervised learning.
- Until reaches a certain level of accuracy

1. Labeled data → training → model(1)
2. Unlabeled data→ model(1)→ new labeled data
3. Labeled data + new labeled data → training → model (2)
4. Unlabeled data → model(2) → new labeled data
5. More data → training→ model(3)
6. …

Co-training



1. Labeled data → training → modelA（1） + modelB (1)
2. Unlabled data → modelA & modelB → labeled Data by modelA & modelB
3. Labeled data + new labeled data → training → modelA(2), modelB(2)
4. More unlabeled data → modelA(2), modelB(2) → labeled data by A,B
5. …

Co-training (cont.)

- *two (or more) classifiers are trained using the same seed set* of labeled examples, but each classifier trains with a disjoint subset of features.
- These feature subsets are commonly referred to as different views that the classifiers have on the training examples.

A.P. hired Mr.Bob as their CFO?

Lexical features
e.g. Capitalization=X
Prefix=Mr.

content features
e.g. contain(CFO)

A co-training algorithm

- features x can be separated into two types x_1,x_2
- either x_1 or x_2 is sufficient for classification – i.e.

there exists functions f_1 and f_2 such that

$$f(x) = f_1(x_1)=f_2(x_2) \text{ has low error}$$

Given:

- a set L of labeled training examples
- a set U of unlabeled examples

Create a pool U' of examples by choosing u examples at random from U

Loop for k iterations:

Use L to train a classifier h_1 that considers only the x_1 portion of x

Use L to train a classifier h_2 that considers only the x_2 portion of x

Allow h_1 to label p positive and n negative examples from U'

Allow h_2 to label p positive and n negative examples from U'

Add these self-labeled examples to L

Randomly choose $2p + 2n$ examples from U to replenish U'

The co-training algorithm

1. Learn a classifier $f_1(x_1)$ from a set of labeled examples L
2. Run the classifier on unlabeled examples
3. Pick some high-confidence predictions and add then to L
4. Repeat steps 1-3 but learn $f_2(x_2)$
5. Start over….

The idea: f_2 is trained on errors made by f_1, which are uncorrelated with f_2's errors.

Active Learning

- Human involved: all examples are labeled **by a human**
- Unlabeled data to be labeled: is **carefully selected** by the machine.

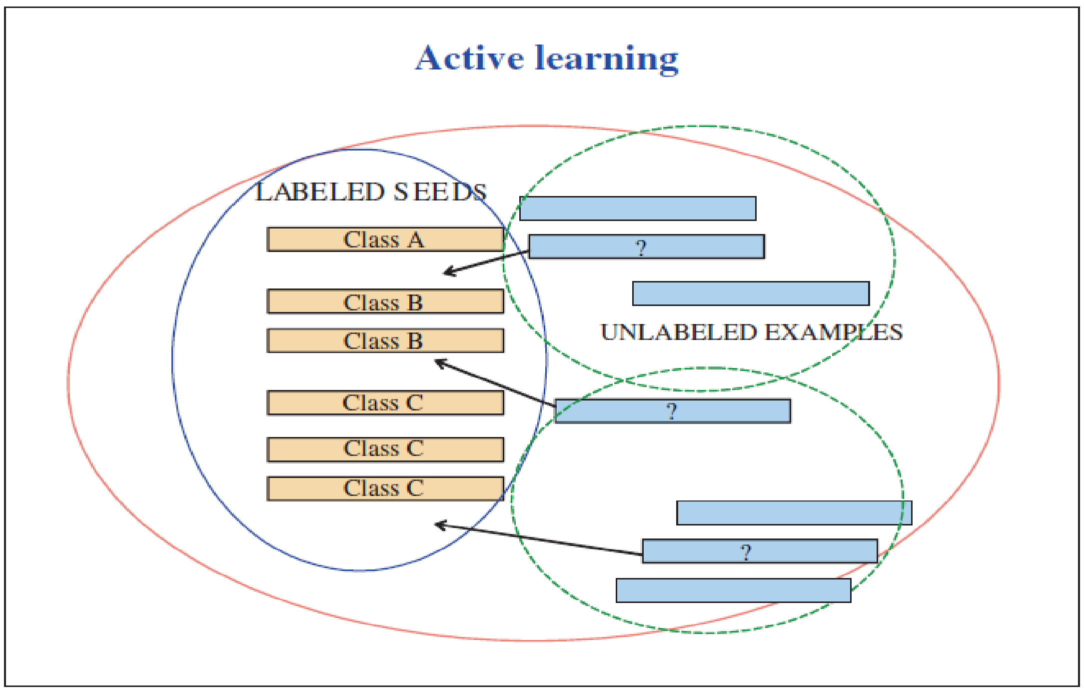


Fig. 6.5. Active learning: Representative and diverse examples to be labeled by humans are selected based on clustering.

Selective examples in active learning

- Most *uncertain* and most *informative*
- *Representative* or *diverse* with the other unlabeled examples

How to select those examples ?

- The probability , Entropy-based measure → *uncertain*
- Similarity calculation, clustering, outliers in the clusters → *representative*

Experiments of supervised and semi-supervised method

- Chinese NP chunking: Experiments with Supervised and semi-supervised learning 国立台湾大学 2008年的文章

	Tag accuracy	Precision	Recall	F-rate
封閉測試 ：测试语料和训练语料同种类型，70%训练，30%测试				
supervised	92.06%	84.65%	86.28%	85.46%
supervised II	91.76%	81.71%	86.05%	83.82%
semi-supervised	92.19%	84.85%	86.64%	85.73%
開放測試 ：测试语料与训练语料完全不同				
supervised	89.03%	67.31%	72.92%	70%
supervised II	83.83%	63.06%	69.23%	66%
semi-supervised	91.61%	76.47%	81.25%	78.79%

When test corpus and training corpus are different.

Summarization

- *Machine learning methods to identify named entities.*
- *Supervised vs. semi-supervised methods*
- *Core learning engines are language independent*
- *Feature extraction relies on language specific properties*

References

- Christopher D.Manning, Hinrich Schuetze, “ Foundation of statistical natural language processing”
- Chapter 5 and chapter 6 of the textbook
- Mark Stevenson and Mark A.Greenwood, “ Comparing Information Extraction Pattern Models”