Booststrapping Methods for Natural Language Processing

DMKM - UPC

2012
Introduction

Degree of supervision
Supervised vs. Unsupervised learning

- **Supervised learning**: learn a mapping from \( x \) to \( y \) given a training set \( \{(x_i, y_i)\} \) of examples \( x_i \) annotated with target labels \( y_i \)
  - The example annotation procedure can be costly in time and in human effort.
  - For NLP tasks thousands of examples can be necessary (i.e. learning models for tagging, parsing, NERC, SRL, IE, etc)
  - For NLP tasks annotating an example can be difficult (i.e. parsing, SRL, IE, etc)

- **Unsupervised learning** (Structural learning): find interesting structure in unnannotated examples.
Weak supervision

- **Bootstrapping methods**: learn a mapping from $x$ to $y$ given a training set $\{(x_i, y_i)\} \cup \{z_i\}$ of few examples $x_i$ annotated with target labels $y_i$ and many unannotated examples $z_i$

  - Try to enlarge the annotated examples $\{(x_i, y_i)\}$ from the unannotated ones $\{z_i\}$ with the most appropriate examples

- **Active learning** within supervised learning, as it requires human supervision. Aim: achieve high accuracy using as few labeled examples as possible

- **Semi-supervised learning** without human supervision. Aim: achieve high accuracy using as few unlabeled examples as possible.
Active Learning

Prototypical algorithm

Query strategy frameworks

Query by uncertainty sampling

Query by committee
Pool-based active learning algorithm

Input: A set $D_L$ of labeled data
       A set $D_U$ of unlabeled data
Output: A classifier $C$

Learn classifier $C$ by applying learner B to $D_L$
repeat
   $I = \text{query_strategy}(C(D_U))$
   $I_L = \text{human\_classification}(I)$
   $D_L = D_L \cup I_L$
   Learn classifier $C$ by applying learner B to $D_L$
until stopping criterion
Stoping criterion

- C performs as randomly selecting I ($D_U$ annotated in advance!!)
- C performs as passively learning C ($D_U$ annotated in advance!!)
- K iterations
- $\text{confidence}(C_i, D'_U) \geq \text{confidence}(C_{i+1}, D'_U)$
Active Learning

Prototypical algorithm

Query strategy frameworks
Query by uncertainty sampling
Query by committee
Query strategy frameworks

- Most extended approaches in NLP:
  - Uncertainty sampling
  - Query-by-committee

- Other approaches:
  - Density-weighted methods, Expected model change, Expected error reduction, Variance reduction
Active Learning

Prototypical algorithm
Query strategy frameworks
Query by uncertainty sampling
Query by committee
Uncertainty sampling

\[ l = \{ x_i^* \mid x_i^* = \arg\min_{x_i \in D_U} \text{certainty}(C(x_i)) \}_{i<K} \]

- Learner B provides a confidence score for each prediction \( C(x) \)
- Examples:
  - Least confident sampling
  - Margin sampling
  - Entropy sampling
Uncertainty sampling

Least confident sampling

\[ x^* = \arg\max_x 1 - P_\phi(\hat{y}|x) \quad \hat{y} = \arg\max_y P_\phi(y|x) \]
Uncertainty sampling

**Least confident sampling**

\[ x^* = \arg\max_x 1 - P_\phi(\hat{y}|x) \quad \hat{y} = \arg\max_y P_\phi(y|x) \]

- Example (Lewis and Gale, 1995)
  - Learning logistic regression models for binary text classification
  - Model:
    \[
    P(C|d) = \frac{e^{(a+b) \sum_{i=1}^k \log \frac{P(d_i|C)}{P(d_i|\overline{C})}}}{1 + e^{(a+b) \sum_{i=1}^k \log \frac{P(d_i|C)}{P(d_i|\overline{C})}}} \quad \text{where } d_i \text{ is a word in } d
    \]
    \[
    (a, b) = \arg\max \prod_d P(C|d)
    \]
  - Query strategy: \( I = \{d \in D_u : P(C|d) \sim 0.5\} \)
Uncertainty sampling

Least confident sampling
\[ x^* = \arg\max_x 1 - P_\phi(\hat{y} | x) \quad \hat{y} = \arg\max_y P_\phi(y | x) \]

- Example (Lewis and Gale, 1995)
  - Learning logistic regression models for binary text classification
  - Model:
    \[ P(C|d) = \frac{e^{(a+b) \sum_{i=1}^{k} \log \frac{P(d_i|C)}{P(d_i|\bar{C})}}}{1 + e^{(a+b) \sum_{i=1}^{k} \log \frac{P(d_i|C)}{P(d_i|\bar{C})}}} \text{ where } d_i \text{ is a word in } d \]
    \[ (a, b) = \arg\max_a, b \prod_{d} P(C|d) \]
  - Query strategy: \( P(C|d) \sim 0.5 \)

- **CONS**: only consider information about the most probable label \( \hat{y} = \arg\max_y P_\phi(y | x) \)

- Query strategy frameworks
- Query by uncertainty sampling
- Query by committee
Uncertainty sampling

Margin sampling

\[ x^* = \arg\min_x P_\phi(\hat{y}_1|x) - P_\phi(\hat{y}_2|x) \]
Uncertainty sampling

Margin sampling

\[ x^* = \arg\min_x P_\phi(\hat{y}_1|x) - P_\phi(\hat{y}_2|x) \]

Example (Scheffer et al., 2001)

- Learn HMMs for Information Extraction
- Model:
  \[ \lambda = (\pi, a, b) \]
  \[ \pi_i = P(q_1 = S_i) \]
  \[ a_{ij} = P(q_{t+1} = S_j|q_t = S_i) \]
  \[ b_i(O_t) = P(O_t|q_t = S_i) \]
  model estimation: Baum – Welch algorithm

Query strategy:

\[ \arg\min_{O} \max_i \{ P(q_t = S_i|O, \lambda) \} - \max_{j \neq i} \{ P(q_t = S_j|O, \lambda) \} \]
Uncertainty sampling

**Margin sampling**

\[ x^* = \arg\min_x P_\phi(\hat{y}_1|x) - P_\phi(\hat{y}_2|x) \]

- Example (Scheffer et al., 2001)
  - Learn HMMs for Information Extraction
  - Model:
    \[ \lambda = (\pi, a, b) \]
    \[ \pi_i = P(q_1 = S_i) \]
    \[ a_{ij} = P(q_{t+1} = S_j|q_t = S_i) \]
    \[ b_i(O_t) = P(O_t|q_t = S_i) \]
  - Model estimation: Baum–Welch algorithm
  - Query strategy:
    \[ \arg\min_{\lambda} \max_{i} \{P(q_t = S_i|O, \lambda) - \max_{j \neq i} \{P(q_t = S_j|O, \lambda)\} \}
    
  - **CONS:** for large label sets, still ignores much information about the output
Uncertainty sampling

Entropy sampling

\[ x^* = \arg\max_x \sum_{i \leq |Y|} P_\phi(y_i|x) \log P_\phi(y_i|x) \]
Uncertainty sampling

Entropy sampling

\[ x^* = \arg\max_x - \sum_{i \leq |Y|} P_\phi(y_i|x) \log P_\phi(y_i|x) \]

▶ Example (Scheffer et al., 2001)
  ▶ Learn HMMs for Information Extraction
  ▶ Model:
    \[ \lambda = (\pi, a, b) \]
    \[ \pi_i = P(q_1 = S_i) \]
    \[ a_{ij} = P(q_{t+1} = S_j|q_t = S_i) \]
    \[ b_i(O_t) = P(O_t|q_t = S_i) \]
  
  model estimation: Baum – Welch algorithm

▶ Query strategy:

\[ \arg\max_{O} - \sum_{i \leq |S|} P(q_t = S_i|O, \lambda) \log P(q_t = S_i|O, \lambda) \]
Active Learning
Prototypical algorithm
Query strategy frameworks
Query by uncertainty sampling
Query by committee
Query-by-committe

\[ C = \{ \phi_1, \phi_2, \ldots, \phi_k \} \] ensemble or committe of models \( \phi_i \)
learned using learner B

\[ I = \{ x_i^* | x_i^* = \arg\max_{x_i \in D_U} \text{disagreement}(C(x_i)) \}_{i<K} \]

- How to generate models \( \phi_i \)?
- How to compute the disagreement \( D_C(x) \)?
Query-by-committee: generation of $\phi_i$

- **Classically**: Randomly sampling $\phi_i$ from some posterior distribution $P(\phi|O)$ (B must be a probabilistic learner)

- **Query by bagging**: Randomly sampling $D_L^i \subset D_L$ to learn $\phi_i$ with learner B (Breiman, 96)

- **Query by boosting**: Iteratively learning weak model $\phi_i$ that focuses on the cases in which models $\{\phi_1...\phi_{i-1}\}$ failed (Freund and Schapire, 97)

- **Co-testing**: Learning $\phi_i$ with a different view $v_i$ of $D_L$ with learner B

How many models in the committee? Depends on the task. Small committee sizes showed to work well in practice.
### Query-by-committee: computation of $D_C(x)$

Examples:

**Margin sampling**

$$x^* = \arg\max_{x} \left( \max_{\phi_i} \text{Conf}(\phi_i(x)) - \min_{\phi_j} \text{Conf}(\phi_j(x)) \right)$$

**Vote Entropy sampling**

$$x^* = \arg\max_{x} \left( -\sum_{i \leq |Y|} \frac{V(y_i, x)}{K} \log \frac{V(y_i, x)}{K} \right)$$
(Muslea et al, 06) Co-testing + margin sampling for Web page classification
Classes = {course homepages (c), others (¬c)}
View1: words in the webpage, View2: words in hyperlinks pointing to it

\[
\text{learn } C = \{NB_{\text{view1}}, NB_{\text{view2}}\} \text{ using } D_L
\]

repeat
\[
\text{select } x = \arg\max_{x \in D_U} \left( \max_{NB_i} \text{Conf}(NB_i(x)) - \min_{NB_j} \text{Conf}(NB_j(x)) \right)
\]
\[
D_U = D_U - \{x\}; \quad D_L = D_L \cup \{<x, y>\}
\]

learn \( C = \{NB_{\text{view1}}, NB_{\text{view2}}\} \) using \( D_L \)

until \( K \) iterations

\[
NB(x) = \arg\max_{y \in \{c, ¬c\}} \sum_{NB_i(x)=y} \text{Conf}(NB_i(x))
\]
Query-by-committee: Examples

(Engelson and Dagan, 99) HMMs + vote entropy sampling for POS tagging

\[ \text{learn } C = \{HMM_1, \ldots, HMM_k\} \text{ using } D_L \text{ to randomly sample their parameters from their approximated posterior distributions} \]

\[ \text{repeat} \]

\[ \text{select } x = \arg\max_{x \in D_U} - \frac{1}{\log k} \sum_{i \leq |Y|} \frac{V(y_i, x)}{k} \log \frac{V(y_i, x)}{k} \]

\[ D_U = D_U - \{x\}; \quad D_L = D_L \cup \{< x, y >\} \]

\[ \text{learn } C = \{HMM_1, \ldots, HMM_k\} \text{ using } D_L \text{ as before} \]

\[ \text{until } K \text{ iterations} \]

\[ \text{MMH}(x) = \arg\max_{y \in Y} \sum_{MHH_i(x) = y} \text{Conf}(HMM_i(x)) \]
Semi-supervised Learning

Semi-supervised learning vs. active learning
Prototypical algorithm
Selection strategy frameworks
Self-training
Co-training
Multiview learning
Difference with active learning

Active learning
▶ exploits what the **human** knows about the labels of unlabeled examples.
▶ exploits what the **learner** knows about difficulty of learning unlabeled examples.

Semi-supervised learning
▶ exploits what the **learner** knows about the labels of unlabeled examples.
Semi-supervised Learning

Semi-supervised learning vs. active learning

Prototypical algorithm

Selection strategy frameworks

Self-training

Co-training

Multiview learning
Semi-supervised learning algorithm

Input: A set $D_L$ of labeled data
A set $D_U$ of unlabeled data
Output: A classifier $C$

Learn classifier $C$ by applying learner $B$ to $D_L$
repeat
  $I_L = \text{selection\_strategy}(C(D_U))$
  $D_L = D_L \cup I_L$
  Learn classifier $C$ by applying learner $B$ to $D_L$
until stopping criterion
Stopping criterion

- $K$ iterations
- $\text{confidence}(C_i, D'_U) \geq \text{confidence}(C_{i+1}, D'_U)$
Semi-supervised Learning

Semi-supervised learning vs. active learning
Prototypical algorithm
Selection strategy frameworks
Self-training
Co-training
Multiview learning
Selection strategy frameworks

- Self-training: analogy with uncertainty sampling in active learning
- Co-training and Multi-view learning: analogy with query-by-committee in active learning
Semi-supervised Learning

Semi-supervised learning vs. active learning
Prototypical algorithm
Selection strategy frameworks
Self-training
Co-training
Multiview learning
Self-training

\[ I_L = \{(x_i^*, C(x_i^*)) \mid x_i^* = \arg\max_{x_i} \text{certainty}(C(x_i))\}_{i<K} \]

Learner B provides a confidence score for each prediction \( C(x_i) \)
Self-training: example (1)

(Mihalcea, 04) Naive Bayes for word sense disambiguation
learn one classifier to disambiguate each particular word occurrence 
\(o\) appearing in a context

- **Model:**
  \[
P(S|o = \langle f_1, ..., f_n \rangle) = \alpha P(S) \prod_{i=1}^{n} P(f_i|S)
  \]

- **Feature set** \(F = \{f_i\} \) (local features): the word itself, POS, words in a window, POS in a window, collocations of \(k\) words in a window, first verb before, etc

- **Selection strategy:** (entropy sampling)
  \[
  \arg\min_o \sum_{i \leq |S|} P(S = s_i|o) \log P(S = s_i|o)
  \]
Self-training: example (2)

(Mihalcea, 04) Naive Bayes for word sense disambiguation
learn one classifier to disambiguate each particular word occurrence
\( o \) appearing in a context

Create \( D'_U \) choosing \( P \) random examples from \( D_U \)
Learn classifier \( C \) by applying NB to \( D_L \)
for \( N \) iterations

\[
I_L = \{ (o_i^*, C(o_i^*)) \mid o_i^* = \arg\min_{o \in D'_U} \sum_{i < |S|} P(s_i|o) \log P(s_i|o) \}
\]

\[
D'_U = D'_U - \{ o | (o, y) \in I_L \} + \Delta D'_U; \quad D_L = D_L \cup I_L
\]
Learn classifier \( C \) by applying NB to \( D_L \)
Semi-supervised Learning

Semi-supervised learning vs. active learning
Prototypical algorithm
Selection strategy frameworks
Self-training
Co-training
Multiview learning
Co-training

\[
\mathbf{C} = \langle \mathbf{C}_1, \mathbf{C}_2 \rangle \text{ pair of classifiers } \mathbf{C}_i \text{ learned using learner B}
\]

\[
\mathbf{I}_L = \langle \mathbf{I}_{L1}, \mathbf{I}_{L2} \rangle \text{ pair of instance sets } \mathbf{I}_L_i \text{ selected at each iteration}
\]

\[
\mathbf{I}_{L1} = \{ (x^*_i, \mathbf{C}_2(x^*_i)) \mid x^*_i = \arg\max_{x_i \in \mathbb{D}_U} \text{certainty} (\mathbf{C}_2(x_i)) \} \quad i < K
\]

\[
\mathbf{I}_{L2} = \{ (x^*_i, \mathbf{C}_1(x^*_i)) \mid x^*_i = \arg\max_{x_i \in \mathbb{D}_U} \text{certainty} (\mathbf{C}_1(x_i)) \} \quad i < K
\]
Co-training

\[ C = \langle C_1, C_2 \rangle \] pair of classifiers \( C_i \) learned using learner B

\[ I_L = \langle I_{L1}, I_{L2} \rangle \] pair of instance sets \( I_{Li} \) selected at each iteration

\[ I_{L1} = \{(x_i^*, C_2(x_i^*)) \mid x_i^* = \arg \max_{x_i \in D_U} \text{certainty}(C_2(x_i))\} \quad i<K \]

\[ I_{L2} = \{(x_i^*, C_1(x_i^*)) \mid x_i^* = \arg \max_{x_i \in D_U} \text{certainty}(C_1(x_i))\} \quad i<K \]

- Each classifier has an independent viewpoint of the data.
Co-training

\[ C = \langle C_1, C_2 \rangle \] pair of classifiers \( C_i \) learned using learner \( B \)

\[ I = \langle I_{L_1}, I_{L_2} \rangle \] pair of instance sets \( I_{L_i} \) selected at each iteration

\[ I_{L_1} = \{(x_i^*, C_2(x_i^*)) \mid x_i^* = \arg\max_{x_i \in D_U} \text{certainty}(C_2(x_i))\} \]

\[ I_{L_2} = \{(x_i^*, C_1(x_i^*)) \mid x_i^* = \arg\max_{x_i \in D_U} \text{certainty}(C_1(x_i))\} \]

- Each classifier has an independent viewpoint of the data.
- Some works use different learners \( B_i \) for each \( C_i \)
Co-training: example (1)

(Mihalcea, 04) Naive Bayes for word sense disambiguation

learn one classifier to disambiguate each particular word occurrence

\( o \) appearing in a context

- **Model**: same as before
- **Feature set** \( F_1 = \{ f_i^1 \} \) *(local features)*: the word itself, POS, words in a window, POS in a window, collocations of \( k \) words in a window, first verb before, etc
- **Feature set** \( F_2 = \{ f_i^2 \} \) *(topical features)*: has a sense keyword in the context (larger than window), has sense keybigram in the context
- **Selection strategy**: same as before
Co-training: example (2)

(Mihalcea, 04) Naive Bayes for word sense disambiguation

\[ D'_U = \{ D'_{U1}, D'_{U2} \} = P \text{ random examples for each } D'_{U_i} \text{ from } D_U \]

Learn \( C = \{ NB_{F1}, NB_{F2} \} \) using \( D_L \)

for \( N \) iterations

\[
I_{L1} = \{(o^*_i, NB_{F2}(o^*_i)) \mid o^*_i = \arg\min_{o \in D'_{U2}} \sum_{i \leq |S|} P(s_i \mid o, NB_{F2}) \log P(s_i \mid o, NB_{F2})\} \]

\[
I_{L2} = \{(o^*_i, NB_{F1}(o^*_i)) \mid o^*_i = \arg\min_{o \in D'_{U1}} \sum_{i \leq |S|} P(s_i \mid o, NB_{F1}) \log P(s_i \mid o, NB_{F1})\} \]

\[
D'_{U_i} = D'_{U_i} - \{ o \mid (o, y) \in I_{L_i} \} + \Delta D'_{U} \]

\[
D_{L_i} = D_{L_i} \cup I_{L_i} \]

Learn \( C = \{ NB_{F1}, NB_{F2} \} \) using \( D_{L1} \) and \( D_{L2} \) respectively

\[
C_{output}(x) = \arg\max_y \sum_{NB_i(x) = y} \text{Conf}(NB_i(x))
\]
Semi-supervised Learning

Semi-supervised learning vs. active learning
Prototypical algorithm
Selection strategy frameworks
Self-training
Co-training
Multiview learning
Multiview learning

- Generalization of co-training

\[ C = \{ C_1, C_2, \ldots, C_n \} \text{ set of classifiers } C_i \text{ learned using learner } B \text{ (could be different learners)} \]

\[ I = \{ (x_i^*, C(x_i^*)) \mid x_i^* = \operatorname{argmax} \text{ agreement}(C(x_i)) \} \text{ for } i < K \]

\[ x_i \in D_u \]

- It is not mandatory for each classifier to provide an independent viewpoint of the data.
References (1)


References (2)