Active learning in sequence labeling

Tomáš Šabata

11. 5. 2017

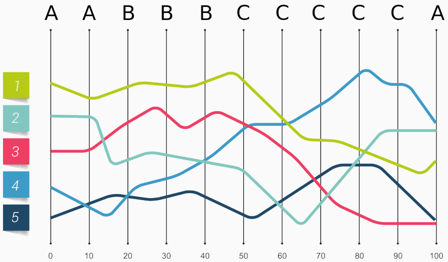
Czech Technical University in Prague
Faculty of Information technology
Department of Theoretical Computer Science

Table of contents

Introduction

Sequence modeling and labeling problem definition

- Sequence of states
- Sequence of observations



Obrázek 1: Sequence representation

Sequence modeling and labeling problem definition

1. Sequence modeling
    - Given a sequence of states/labels and sequence of observations, find a model that the most likely generates the sequences.
2. Sequence labeling
    - Given a sequence of observations, determine an appropriate label/state for each observation.

- Reducing errors by considering relations.

Applications

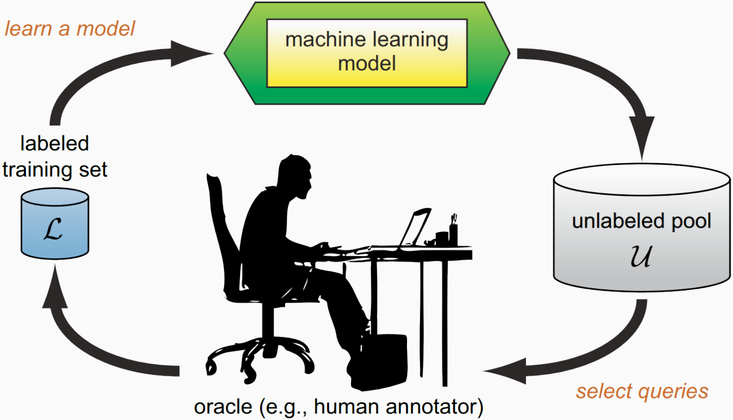
- Handwriting recognition
- Facial expression dynamic modeling
- DNA analysis
- Part-of-speech tagging
- Speech recognition
- Video analysis

Active learning

What is active learning?

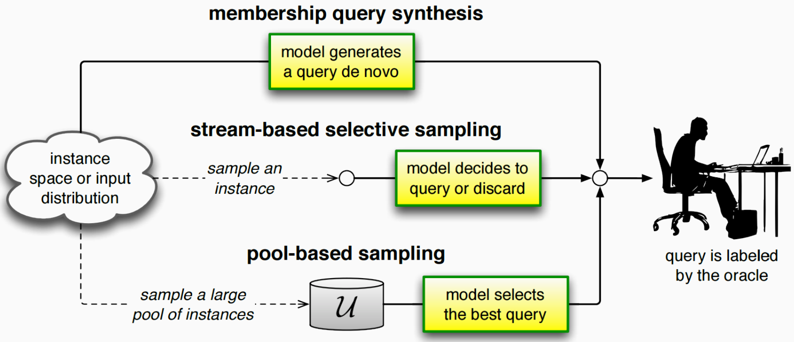
- *The quality of labels makes a huge difference. Garbage in, garbage out.*
- *Obtaining "golden"annotation data can be really expensive.*

What is active learning?



Obrázek 2: Active learning cycle

Scenarios



Obrázek 3: Active learning scenarios

AL algorithm

Data:

L - set of labeled examples

U - set of umlabeled examples

θ - utility function

while *stopping criterion is not met* **do**

1. learn model M from L;
2. for all $x_i \in U : u_{x_i} \leftarrow \theta_M(x_i)$;
3. select example $x^* \in U$ with the highest utility function u_i;
4. query annotator for label of example x^*;
5. move $< y, x^* >$ to L;

end

return L

Algorithm 1: General pool-based AL framework

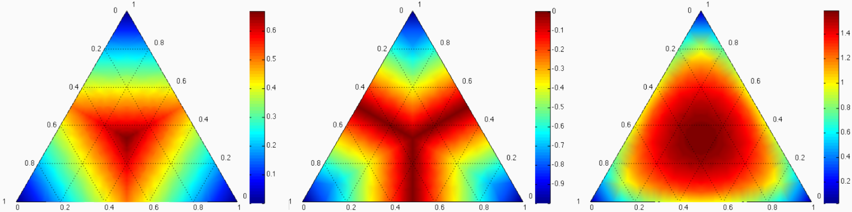
Query strategies frameworks

1. Uncertainty Sampling
2. Query-By-Committee
3. Expected Model Change
4. Expected Error Reduction
5. Variance Reduction
6. Density-Weighted Methods

Frameworks: Uncertainty Sampling

- Simplest, most commonly used
- Intuitive for probabilistic learning model
- Binary problems: choose intance with posterior probability near to 0.5
- Multiclass problems:
  - Least confident - $x^*_{LC} = \arg\max_x 1 - P_\theta(\hat{y}|x)$
  - Margin sampling - $x^*_M = \text{argmin}_x P_\theta(\hat{y}_1|x) - P_\theta(\hat{y}_2|x)$
  - Entropy - $x^*_H = argmax_x - \sum_i P_\theta(y_i|x)\text{log}P_\theta(y_i|x)$

Frameworks: Uncertainty Sampling



(a) least confident (b) margin (c) entropy

Obrázek 4: Uncertainty sampling for three-class classification problem

- Application dependent
- Entropy - minimizing log-loss
- LC + Margin - minimizing classification error

Frameworks: Query-By-Committee

- We maintain a committee $\mathcal{C} = \{\theta_1, ..., \theta_C\}$ of models trained on $\mathcal{L}$
- The most informative query is considered to be the instance about which they most disagree.
- We need to ensure variability of models in the beginning
- Measure of disagreement:
  - Vote entropy - $x^*_{VE} = argmax_x - \sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C}$
  - Kullback-Leibler divergence - $x^*_{KL} = argmax_x \frac{1}{C} \sum_{c=1}^{C} D_{KL}(P_{\theta^{(c)}} || P_C)$

Frameworks: Expected Model Change

- Capable for models using gradient based training.
- Query instance which would cause the largest model change.
- Use a gradient of the objective function $\bigtriangledown \ell_\theta(\mathcal{L})$
- $x^*_{EMC} = \text{argmax}_x \sum_i P_\theta(y_i|x) \parallel \bigtriangledown \ell_\theta(\mathcal{L} \cup < y_i, x >) \parallel$
- Note: $\parallel \bigtriangledown \ell_\theta(\mathcal{L}) \parallel$ should be close to zero therefore we can use an approximation $\parallel \bigtriangledown \ell_\theta(\mathcal{L} \cup < y_i, x >) \parallel \approx \parallel \bigtriangledown \ell_\theta(< y_i, x >) \parallel$

Frameworks: Expected Error Reduction

- Estimate the expected future error of a model trained on $\mathcal{L} \cup < x, y >$
- Methods:
  - Minimizing the excepted 0/1-loss
    $x^* = \text{argmin}_x \sum_i P_\theta(y_i|x)\Big(\sum_{u=1}^{U} 1 - P_{\theta + <x,y_i>}(\hat{y}|x^{(u)})\Big)$
  - Minimizing the excepted log-loss
    $x^* = \text{argmin}_x \sum_i P_\theta(y_i|x)\Big(- \sum_{u=1}^{U} \sum_j P_{\theta + <x,y_i>}(y_j|x^{(u)}) \log P_{\theta + <x,y_i>}(y_j|x^{(u)})\Big)$
- In most cases the most computationally expensive query framework
  - Logistic regression - $\mathcal{O}(ULG)$
  - CRF - $\mathcal{O}(TM^{T+2}ULG)$

Frameworks: Variance Reduction

- Use the bias-variance decomposition
- $E_T[(\hat{y} - y)^2|x] =$
  $E[(y - E[y|x])^2] + (E_{\mathcal{L}}[\hat{y}] - E[y|x])^2 + E_{\mathcal{L}}[(\hat{y} - E_{\mathcal{L}}[\hat{y}])^2]$
- Model dependent framework

Frameworks: Density-Weighted Methods

- Informative instances should not only be those which are uncertain, but also those which are "representative" of the underlying distribution
- Uses one of other query strategies as base query strategy (e.g. uncertainty sampling)
- $x^* = argmax_x \phi_A(x) \times \left(\frac{1}{U} \sum_{u=1}^{U} sim(x, x^{(u)}) \right)^{\beta}$
- The metod is more robust to outliers in dataset.

Active learning problem variants

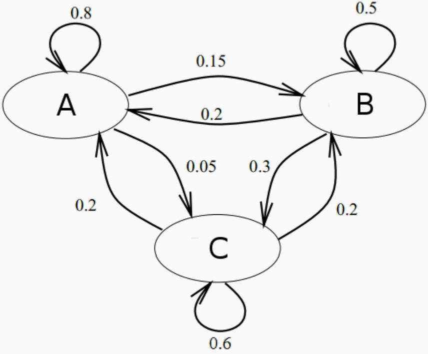
- Active Learning for Structured Outputs
  - Instance is not represented by a single feature vector, but rather a structure.
  - e.g.: Sequences, trees, grammars.
- Active Feature Acquisition
  - Selection of salient unused features
  - e.g.: Medical tests, sensitive information
- Active Class Selection
  - Learner is allowed to query a known class label, and obtaining each instance incurs a cost.
- Active Clustering
  - Generate (or subsample) instances in such a way that they self-organize into groupings
  - Try to get less overlap or noise than with random sampling

Active learning problem variants

- Active Learning for Structured Outputs
  - Instance is not represented by a single feature vector, but rather a structure.
  - e.g.: Sequences, trees, grammars.
- Active Feature Acquisition
  - Selection of salient unused features
  - e.g.: Medical tests, sensitive information
- Active Class Selection
  - Learner is allowed to query a known class label, and obtaining each instance incurs a cost.
- Active Clustering
  - Generate (or subsample) instances in such a way that they self-organize into groupings
  - Try to get less overlap or noise than with random sampling

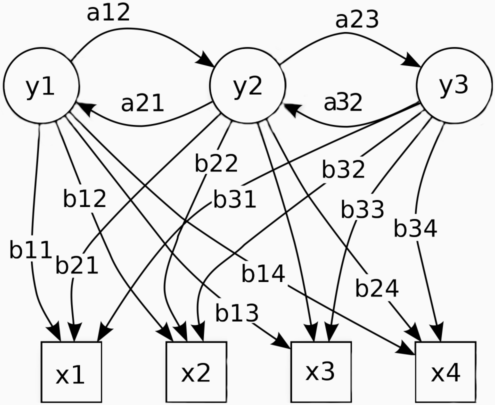
Graphical models

Models: Markov model



Obrázek 5: Markov model/chain

Models: Hidden Markov model



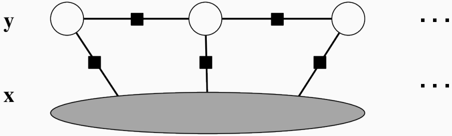
Obrázek 6: Hidden Markov model

Models: Hidden Markov model

$\lambda = (A, B, \pi)$

- Set of hidden states $Y = \{y_1, y_2, ..., y_N\}$, set of observable values $X = \{x_1, x_2, ..., x_M\}$
- Sequence of states $Q = q_1 q_2 q_3 ... q_T$ sequence of outputs $O = o_1 o_2 o_3 ... o_T$
- Transition probability matrix $A = \{a_{ij}\}$
  $a_{ij} = P(q_t = y_j | q_{t-1} = y_i), \quad 1 \leq i, j \leq N$
- Emission probability distribution $B = \{b_{i,j}\}$
  $b_{i,j} = P(o_t = x_j | q_t = y_i), \quad 1 \leq i \leq N, \quad 1 \leq i \leq M$
- Initial probability distribution $\pi = \{\pi_i\}$
  $\pi_i = P(q_1 = y_i), \quad 1 \leq i \leq N$

Models: Conditional random field (linear chain)



Obrázek 7: Linear chain conditional random field

- Discriminative model $P(Y|X)$, we do not explicitly model $P(X)$.
- Perform better than HMMs when the true data distribution has higher-order dependencies than the model.
- $P(Y|X) = \frac{1}{Z(x)} \prod_{t=1}^{T} \exp\Big(\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t) \Big)$
- $Z(X) = \sum_{Y} \prod_{t=1}^{T} \exp\Big(\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t) \Big)$

Models: Summary

1. HMM
   - $P(Q,O) \propto \prod_{t=1}^{T} P(q_t|q_{t-1})\, P(o_t|q_t)$

---

2. CRF
   - $$P(Q|O) \propto \frac{1}{Z_O} \prod_{t=1}^{T} \exp\left(\sum_j \lambda_j f_j(q_t, q_{t-1}) + \sum_k \mu_k g_k(q_t, o_t) \right)$$

Active learning in sequence labeling

Uncertainty sampling

- Least confident
  - $x^*_{LC} = \text{argmax}_x\, 1 - P_\theta(\hat{y}|x)$
  - Viterbi path
- Margin sampling
  - $x^*_M = \text{argmin}_x\, P_\theta(\hat{y}_1|x) - P_\theta(\hat{y}_2|x)$
  - N-best aglorithm
- Entropy
  - Token entropy - $x^*_{TE} = \text{argmax}_x -\frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{M} P_\theta(y_t = m)\log P_\theta(y_t = m)$
  - Total token entropy
  - Sequence entropy $x^*_{SE} = \text{argmax}_x -\sum_{\hat{y}} P_\theta(\hat{y}|x)\log P_\theta(\hat{y}|x)$
  - N-best sequence entropy $x^*_{SE} = \text{argmax}_x -\sum_{\hat{y}\in\mathcal{N}} P_\theta(\hat{y}|x)\log P_\theta(\hat{y}|x)$

Query by Committee

- Query-by-bagging (each model has unique modified set $\mathcal{L}^{(c)}$)
- Vote entropy - disagreement over Viterbi's paths
  $x^*_{VE} = \text{argmax}_x - \frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{M}\frac{V(y_t,m)}{C}\log\frac{V(y_t,m)}{C}$
- Kullback Leibler
  $x^*_{KL} = \text{argmax}_x \frac{1}{T}\sum_{t=1}^{T}\frac{1}{C}\sum_{c=1}^{C} D_{KL}(\theta^{(c)}||\mathcal{C})$
  $D(\theta^{(c)}||\mathcal{C}) = \sum_{m=1}^{M} P_{\theta^{(c)}}(y_t = m) log\frac{P_{\theta^{(c)}}(y_t=m)}{P_{\mathcal{C}}(y_t=m)}$
- Non-normalized variants
- Sequence vote entropy
  $x^*_{SVE} = \text{argmax}_x - \sum_{\hat{y}\in\mathcal{N}^{\mathcal{C}}} P(\hat{y}|x,\mathcal{C})\log P(\hat{y}|x,\mathcal{C})$
- Sequence Kullback-Leibler $x^*_{SKL} = \text{argmax}_x \frac{1}{C}\sum_{c=1}^{C}\sum_{\hat{y}\in\mathcal{N}^{\mathcal{C}}} log\frac{P_{\theta^{(c)}}(\hat{y}|x)}{P_{\mathcal{C}}(\hat{y}|x)}$

Expected Gradient Length

- Exceptation over the N-best labelings.
- $x^*_{EMC} = \text{argmax}_x \sum_{\hat{y} \in \mathcal{N}^C} P_\theta(\hat{y}|x) \parallel \triangledown \ell_\theta(\mathcal{L} \cup < y_i, x >) \parallel$

Expected Error Reduction

- $\mathcal{O}(TM^{T+2}ULG)$ too expensive :(

Density-Weighted Methods

- Information density
  - solves problem that US and QBC are prone to querying outliers
- We need a distance measure for sequences.
  - Kullback-Leibler
  - Euclidean distance
  - Cosine distance
- Drawback: number of required similarity calculations grows quadratically with the number of instances in $\mathcal{U}$.
- Solution: Precompute them.

Semi-supervized active learning in sequence labeling

Fully supervized vs semi-supervized

- FuSAL:
  - Sequence is handled as a whole unit.
  - Sequence-wise vs. token-wise utility functions
- SeSAL
  - Some subsequences can be easily labelled automatically.
  - Decrease labelling effort.
  - Usage of self-training principle.

Fully supervized general algorithm

Data:

B - number of examples to be selected

L - set of labeled examples

U - set of umlabeled examples

θ - utility function

while *stopping criterion is not met* **do**

1. learn model M from L;
2. for all $x_i \in U : u_{x_i} \leftarrow \theta_M(x_i)$;
3. select B examples $x_i \in U$ with highest utility function u_i;
4. annotate sequences using M;
5. query for labels of non-confidential tokens;
6. move newly annotated examples to L;

end

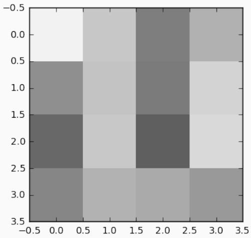
return L

Algorithm 2: General AL framework

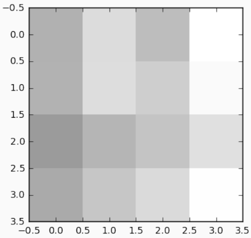
Experiment

Problem definition

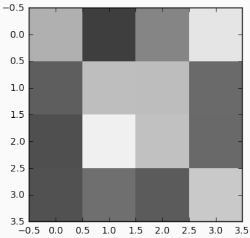
- "Handwritten" letters recognition.
- Each letter is randomly written in one of 6 fonts
- Downscaled to 4x4 pixels
- Letters are orginized in real sentences.



Obrázek 8: A



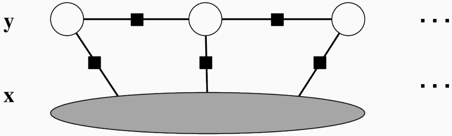
Obrázek 9: B



Obrázek 10: C

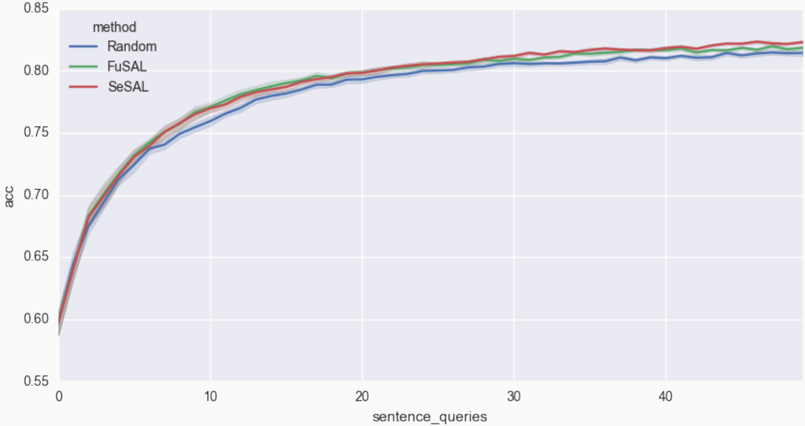
Model and training

- Linear chain conditional random fields.
- 3 labeled sentences in train dataset in the begining .
- Labeled sentences are added to the dataset interatively (50 times).
    - Random choice
    - FuSAL (Least Confident)
    - SeSAL (Least Confident + marginal probability)



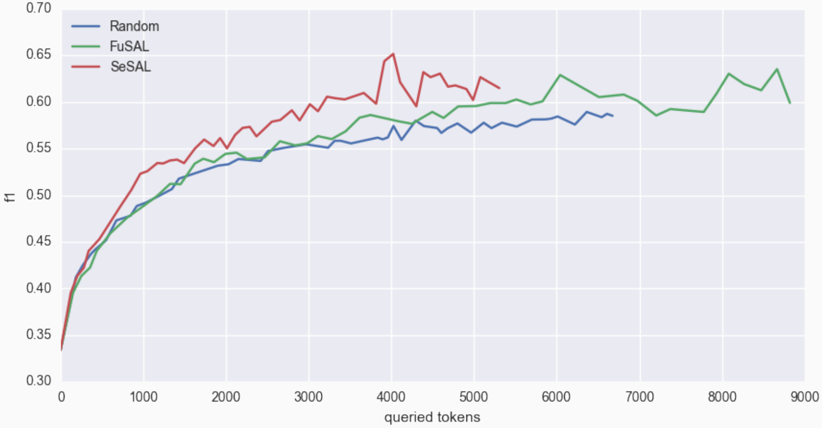
Results

Does it even worth it?

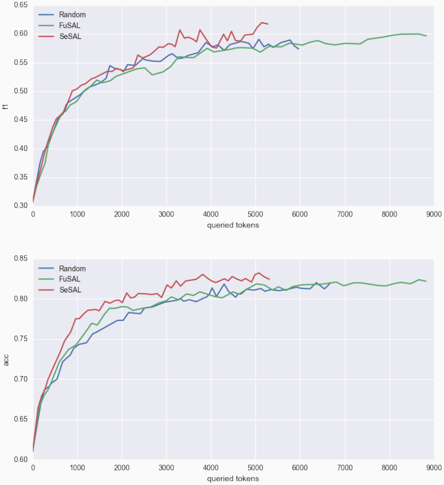


Results

Let's look from another point of view.



Results



Summary

Summary

- It does not work in all cases.
- It can be implementation overhead.

- In the most of cases the active learning helps.
- It can be applied to different structures like sequences or trees
- The combination with semi-supervized learning can lead to rapid save of costs.

- Future work: Different query costs, automatic threshold finding, CT-HMM.

Thank you for your attention.

Questions?