Natural Language Processing and Information Retrieval

Part II: Structured Output

Alessandro Moschitti

Department of information and communication technology
University of Trento
Email: moschitti@dit.unitn.it
Output Label Sets
Simple Structured Output

- We have seen methods for: binary Classifier or multiclassifier single label
- Multiclass-Multilabel is a structured output, i.e. a label subset is output
From Binary to Multiclass classifiers

- Three different approaches:

- **ONE-vs-ALL (OVA)**
  - Given the example sets, \{E1, E2, E3, \ldots\} for the categories: \{C1, C2, C3,\ldots\} the binary classifiers: \{b1, b2, b3,\ldots\} are built.
  
  - For b1, E1 is the set of positives and E2 \cup E3 \cup \ldots is the set of negatives, and so on
  
  - For testing: given a classification instance x, the category is the one associated with the maximum margin among all binary classifiers
From Binary to Multiclass classifiers

- **ALL-vs-ALL (AVA)**
  - Given the examples: \{E1, E2, E3, \ldots\} for the categories \{C1, C2, C3, \ldots\}
  - build the binary classifiers:
    \{b_{1_2}, b_{1_3}, \ldots, b_{1_n}, b_{2_3}, b_{2_4}, \ldots, b_{2_n}, \ldots, b_{n-1_n}\}
  - by learning on E1 (positives) and E2 (negatives), on E1 (positives) and E3 (negatives) and so on...

- For testing: given an example x,
  - all the votes of all classifiers are collected
  - where \(b_{E1E2} = 1\) means a vote for C1 and \(b_{E1E2} = -1\) is a vote for C2

- Select the category that gets more votes
From Binary to Multiclass classifiers

**Error Correcting Output Codes (ECOC)**

- The training set is partitioned according to binary sequences (codes) associated with category sets.
  - For example, 10101 indicates that the set of examples of C1, C3 and C5 are used to train the C_{10101} classifier.
  - The data of the other categories, i.e. C2 and C4 will be negative examples.

- **In testing:** the code-classifiers are used to decode one the original class, e.g.
  - $C_{10101} = 1$ and $C_{11010} = 1$ indicates that the instance belongs to C1.
  - That is, the only one consistent with the codes.
Designing Global Classifiers

- Each class has a parameter vector \((w_k, b_k)\)
- \(x\) is assigned to class \(k\) iff
  \[
  w_k^\top x + b_k \geq \max_j w_j^\top x + b_j
  \]
- For simplicity set \(b_k=0\)
  (add a dimension and include it in \(w_k\))
- The goal (given separable data) is to choose \(w_k\) s.t.
  \[
  \forall (x^i, y^i), \quad w_{y^i}^\top x^i \geq \max_j w_j^\top x^i
  \]
Multi-class SVM

Primal problem: QP

\[
\begin{align*}
\min_{w_1, \ldots, w_K} & \quad \frac{1}{2}\|(w_1, \ldots, w_K)\|^2 + C \sum_{ik} \xi_{ik} \\
\text{s.t.} & \quad \forall (i, k), \quad w_{y_i}^\top x^i - w_k^\top x^i \geq 1_{k \neq y^i} - \xi_{ik}
\end{align*}
\]
Structured Output Model

- Main idea: Define scoring function which decomposes as sum of features scores $k$ on “parts” $p$:

\[
score(x, y, w) = w^\top \Phi(x, y) = \sum_{k, p} w_k^\top \phi_k(x_p, y_p)
\]

- Label examples by looking for max score:

\[
prediction(x, w) = \arg\max_{y \in \mathcal{Y}(x)} score(x, y, w)
\]

- Parts = nodes, edges, etc.

  space of feasible outputs
Structured Perceptron

Inputs: Training set \((x_i, y_i)\) for \(i = 1 \ldots n\)

Initialization: \(W = 0\)

Define: \(F(x) = \arg\max_{y \in \text{GEN}(x)} \Phi(x, y) \cdot W\)

Algorithm: For \(t = 1 \ldots T, i = 1 \ldots n\)
\[z_i = F(x_i)\]
If \((z_i \neq y_i)\) \(W = W + \Phi(x_i, y_i) - \Phi(x_i, z_i)\)

Output: Parameters \(W\)
(Averaged) Perceptron

For each datapoint $x^i$

**Predict:**
\[
\hat{y}_i = \arg \max_{y \in \mathcal{Y}} w_t^\top \Phi(x^i, y)
\]

**Update:**
\[
w_{t+1} = w_t + \alpha \left( \Phi(x, y^i) - \Phi(x^i, \hat{y}_i) \right)
\]
update if $\hat{y}_i \neq y^i$

**Averaged perceptron:**
\[
\bar{w} = \frac{1}{T} \sum_{t=1}^{T} w_t
\]
Example: multiclass setting

**Predict:**
\[ \hat{y}_i = \arg \max_y w_y^\top x_i \]

**Update:** if \( \hat{y}_i \neq y_i \) then
\[ w_{y_i,t+1} = w_{y_i,t} + \alpha x_i \]
\[ w_{\hat{y}_i,t+1} = w_{\hat{y}_i,t} - \alpha x_i \]

**Feature encoding:**
\[ \Phi(x_i, y = 1)^\top = [x_i^\top 0 \ldots 0] \]
\[ \Phi(x_i, y = 2)^\top = [0 x_i^\top \ldots 0] \]
\[ \vdots \]
\[ \Phi(x_i, y = K)^\top = [0 0 \ldots x_i^\top] \]
\[ w^\top = [w_1^\top w_2^\top \ldots w_K^\top] \]

**Predict:**
\[ \hat{y}_i = \arg \max_{y \in \mathcal{Y}} w_t^\top \Phi(x_i, y) \]

**Update:**
\[ w_{t+1} = w_t + \alpha \left( \Phi(x, y^i) - \Phi(x_i, \hat{y}_i) \right) \]
update if \( \hat{y}_i \neq y_i \)
Output of Ranked Example List
Support Vector Ranking

Given two examples we build one example \((x_i, x_j)\) whose the main characteristics are:

- The constraint \(y_i (\mathbf{w} \cdot x_i + b) \geq 1 - \xi_i\) allows the point \(x_i\) to violate the hard constraint of Problem 2.13 of a quantity equal to \(\xi_i\). This is clearly shown by the outliers in Figure 2.14, e.g. \(x_i\).

- If a point is misclassified by the hyperplane then the slack variable assumes a value larger than 1. For example, Figure 2.14 shows the misclassified point \(x_i\) and its associated slack variable \(\xi_i\) which is necessarily > 1. Thus, \(\sum_{i=1}^{m} \xi_i\) is an upper bound to the number of errors. The same property is held by the quantity, \(\sum_{i=1}^{m} \xi_i^2\), which can be used as an alternative bound.

- The constant \(C\) tunes the trade-off between the classification errors and the margin. The higher \(C\) is, the lower number of errors the optimal solution commits. For \(C \to \infty\), Problem 2.22 approximates Problem 2.13.

- Similarly to the hard margin error probability upper bound, it can be proven that minimizing \(\frac{1}{2}||\mathbf{w}|| + C \sum_{i=1}^{m} \xi_i^2\) minimizes the error probability of classifiers which are not perfectly consistent with the training data, e.g. they do not necessarily classify correctly all the training data.

\[
\begin{cases}
\min \frac{1}{2}||\mathbf{w}|| + C \sum_{i=1}^{m} \xi_i^2 \\
y_k (\mathbf{w} \cdot (\mathbf{x}_i - \mathbf{x}_j) + b) \geq 1 - \xi_k, \quad \forall i, j = 1, .., m \\
\xi_k \geq 0, \quad k = 1, .., m^2
\end{cases}
\]

\(y_k = 1\) if \(rank(\mathbf{x}_i) > rank(\mathbf{x}_j)\), 0 otherwise, where \(k = i \times m + j\)

- Given two examples we build one example \((x_i, x_j)\)
Concept Segmentation and Classification task

- Given a transcription, i.e. a sequence of words, chunk and label subsequences with concepts
- Air Travel Information System (ATIS)
  - Dialog systems answering user questions
  - Conceptually annotated dataset
  - Frames
An example of concept annotation in ATIS

- User request: *list TWA flights from Boston to Philadelphia*

- The concepts are used to build rules for the dialog manager (e.g. actions for using the DB)
  - from location
  - to location
  - airline code

```
list TWA flights from Boston to Philadelphia

null airline_code null null fromloc.city null toloc.city
```

```
list flights from boston to Philadelphia
FRAME: FLIGHT
FROMLOC.CITY = boston
TOLOC.CITY = Philadelphia
```
Our Approach
(Dinarelli, Moschitti, Riccardi, SLT 2008)

- Use of Finite State Transducer to generate word sequences and concepts
- Probability of each annotation
  \[ \Rightarrow m \text{ best hypothesis can be generated} \]
- Idea: use a discriminative model to choose the best one
  - Re-ranking and selecting the top one
## Experiments

- Luna projects’ Corpus Wizard of OZ

<table>
<thead>
<tr>
<th>Corpus LUNA</th>
<th>Training set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>words</td>
<td>concepts</td>
</tr>
<tr>
<td>Dialogs</td>
<td>183</td>
<td></td>
</tr>
<tr>
<td>Turns</td>
<td>1,019</td>
<td></td>
</tr>
<tr>
<td>Tokens</td>
<td>8,512</td>
<td>2,887</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>1,172</td>
<td>34</td>
</tr>
<tr>
<td>OOV rate</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Re-ranking Model

- The FST generates the most likely concept annotations.
- These are used to build annotation pairs, $\langle s^i, s^j \rangle$. Positive instances if $s^i$ more correct than $s^j$.
- The trained binary classifier decides if $s^i$ is more accurate than $s^j$.
- Each candidate annotation $s^i$ is described by a word sequence where each word is followed by its concept annotation.
Re-ranking framework
Example

- I have a problem with the network card now

\[ S^i : I \textbf{NULL} \text{ have \textbf{NULL} a \textbf{NULL}} \text{ problem } \textbf{PROBLEM-B} \text{ with \textbf{NULL} my \textbf{NULL} monitor } \textbf{HW-B} \]

\[ S^i : I \textbf{NULL} \text{ have \textbf{NULL} a \textbf{NULL}} \text{ problem } \textbf{HW-B} \text{ with \textbf{NULL} my \textbf{NULL} monitor } \]
Flat tree representation

ROOT

NULL  PROBLEM-B  PROBLEM-I  HW-B  HW-I

| Ho | un | problema | col | monitor |
Multilevel Tree

ROOT

NULL

PROBLEM

HW

Ho

PROBLEM-B

problem

un

PROBLEM-I

HW-B

col

HW-I

monitor
Enriched Multilevel Tree
### Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Concept Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVMs</td>
<td>26.7</td>
</tr>
<tr>
<td>FSA</td>
<td>23.2</td>
</tr>
<tr>
<td>FSA+Re-Ranking</td>
<td>16.01</td>
</tr>
</tbody>
</table>

≈ 30% of error reduction of the best model
Structured Perceptron

**Inputs:** Training set \((x_i, y_i)\) for \(i = 1 \ldots n\)

**Initialization:** \(\mathbf{W} = 0\)

**Define:**

\[
F(x) = \arg \max_{y \in \text{GEN}(x)} \Phi(x, y) \cdot \mathbf{W}
\]

**Algorithm:** For \(t = 1 \ldots T, i = 1 \ldots n\)

\[
z_i = F(x_i)
\]

If \((z_i \neq y_i)\)

\[
\mathbf{W} = \mathbf{W} + \Phi(x_i, y_i) - \Phi(x_i, z_i)
\]

**Output:** Parameters \(\mathbf{W}\)
References


References


References

References


References


References


References

- AN INTRODUCTION TO SUPPORT VECTOR MACHINES (and other kernel-based learning methods)
  N. Cristianini and J. Shawe-Taylor Cambridge University Press


The Impact of SSTK in Answer Classification

![Graph showing F1-measure over j]

- $Q(BOW) + A(BOW)$
- $Q(BOW) + A(PT,BOW)$
- $Q(PT) + A(PT,BOW)$
- $Q(BOW) + A(BOW,PT,PAS)$
- $Q(BOW) + A(BOW,PT,PAS)_N$
- $Q(PT) + A(PT,BOW,PAS)$
- $Q(BOW) + A(BOW,PAS)$
- $Q(BOW) + A(BOW,PAS)_N$
Mercer’s conditions (1)

Def. B.11 Eigen Values
Given a matrix $A \in \mathbb{R}^m \times \mathbb{R}^n$, an eigenvalue $\lambda$ and an eigenvector $\bar{x} \in \mathbb{R}^n - \{\bar{0}\}$ are such that

$$A\bar{x} = \lambda\bar{x}$$

Def. B.12 Symmetric Matrix
A square matrix $A \in \mathbb{R}^n \times \mathbb{R}^n$ is symmetric iff $A_{ij} = A_{ji}$ for $i \neq j, i = 1, \ldots, m$ and $j = 1, \ldots, n$, i.e. iff $A = A'$.

Def. B.13 Positive (Semi-) definite Matrix
A square matrix $A \in \mathbb{R}^n \times \mathbb{R}^n$ is said to be positive (semi-) definite if its eigenvalues are all positive (non-negative).
Mercer’s conditions (2)

Proposition 2.27 (Mercer’s conditions)
Let $X$ be a finite input space with $K(\bar{x}, \bar{z})$ a symmetric function on $X$. Then $K(\bar{x}, \bar{z})$ is a kernel function if and only if the matrix

$$ k(\bar{x}, \bar{z}) = \phi(\bar{x}) \cdot \phi(\bar{z}) $$

is positive semi-definite (has non-negative eigenvalues).

- If the Gram matrix: $G = k(\bar{x}_i, \bar{x}_j)$
  is positive semi-definite there is a mapping $\phi$ that produces the target kernel function
The lexical semantic kernel is not always a kernel

- It may not be a kernel so we can use $M' \cdot M$, where $M$ is the initial similarity matrix.

**Proposition B.14** Let $A$ be a symmetric matrix. Then $A$ is positive (semi-) definite iff for any vector $\vec{x} \neq 0$

$$\vec{x}' A \vec{x} > \lambda \vec{x} \quad (\geq 0).$$

From the previous proposition it follows that: If we find a decomposition $A$ in $M' \cdot M$, then $A$ is semi-definite positive matrix as

$$\vec{x}' A \vec{x} = \vec{x}' M' M \vec{x} = (M \vec{x})'(M \vec{x}) = M \vec{x}' \cdot M \vec{x} = ||M \vec{x}||^2 \geq 0.$$
Efficient Evaluation (1)

- In [Taylor and Cristianini, 2004 book], sequence kernels with weighted gaps are factorized with respect to different subsequence sizes.
- We treat children as sequences and apply the same theory

\[
\Delta(n_1, n_2) = \mu(\lambda^2 + \sum_{p=1}^{l,m} \Delta_p(c_{n_1}, c_{n_2}))
\]

Given the two child sequences \(s_1a = c_{n_1}\) and \(s_2b = c_{n_2}\) (\(a\) and \(b\) are the last children), \(\Delta_p(s_1a, s_2b) = \)

\[
\Delta(a, b) \times \sum_{i=1}^{\left|s_1\right|} \sum_{r=1}^{\left|s_2\right|} \lambda^{\left|s_1\right|-i+\left|s_2\right|-r} \times \Delta_{p-1}(s_1[1 : i], s_2[1 : r])
\]
Theory

- Kernel Trick
- Kernel Based Machines
- Basic Kernel Properties
- Kernel Types