Using Salience to Segment Desktop Activity into Projects

Daniel Lowd
University of Washington
<lowd@cs.washington.edu>

Nicholas Kushmerick
Decho Corporation
<nicholask@decho.com>

BACKGROUND: TaskTracer and SmartDesktop

TaskTracer and corporate spin-off SmartDesktop improve knowledge worker productivity by associating each desktop action with a project and using this information for time tracking, interruption recovery, and information retrieval.

OUR GOAL: Automatically infer the project for each action.

Previous work used generic methods that only considered content and compartmented for poor accuracy by skipping predictions when confidence was low.

We present novel "salience" features that explicitly take into account both context and content. Using these features, we beat a finely tuned expert system.

FEATURES: Salience Links Content to Context

Resource Features describe the document or resource associated with the current action.

General: • Full URI • URI subpath • Title words • Body words • Type

Email Only: • Sender • Recipients • Thread-ID

Attachments: • Subject • Recipients • Sender

Past Project Features indicate the project labels of the last four actions.

Salience Features compare the current action to recent actions. Each gives the number of resource features of a given type last seen with a given project.

Shared Salience Features share salience features across all projects, allowing generalization to new projects or users.

Example: Salience Features

\[ W(x,A) = \{\text{"URI subpath A"} = 1, \text{"Body words A"} = 3\} \]

Example: Shared Salience Features

\[ W(x,A) = \{\text{"URI subpath A"} = 1, \text{"Body words A"} = 3\} \]

ALGORITHMS: Four Standard Approaches

Naïve Bayes (NB)

Assume observed features \( W(x) \) are independent given class label (project). The probability is a linear model. Weights are log conditional probabilities.

\[ p(j|x) = \frac{p(x|j)p(j)}{\sum_k p(x|k)p(k)} \]

\[ \log p(j|x) = \Psi(x,j) \]

\[ \log p(j|x) = \log \sum_k e^{\Psi(x,k)} + \log k \]

PRO: Simple and fast. Often surprisingly effective.

CON: Overly strong assumptions.

Passive-Aggressive (PA)

After each example, update weights so that hinge loss \( l \) on most recent example is zero.

\[ w_{n+1} = w_n + \frac{l_n}{\sum_{t=0}^{n} l_t} \Phi(x_n,p_n) \]

\[ l_n = \max(0,1-w_n \cdot \Psi(x_n,p_n)) \]

PRO: Simple and fast. Adapts quickly to new information. Less constrained than NB.

CON: May "forget" what it learned.

Logistic Regression (LR)

Probability is weighted exponential sum:

\[ p(j|x) = \frac{1}{1+e^{-\Psi(x,j)}} \]

\[ \log p(j|x) = \log \sum_k e^{\Psi(x,k)} + \log k \]

PRO: Less constrained than NB.

CON: More prone to overfitting.

Support Vector Machines (SVM)

Choose weights to minimize magnitude of the weight vector and hinge loss:

\[ h(w) = \sum_{t} l_t \]

\[ l_t = \max(0,1-w \cdot \Psi(x_t,p_t)) \]

PRO: Longer memory than PA.

CON: Slower training.

EXPERIMENTS: SVMs Beat Finely Tuned Expert Systems

We evaluated our features and algorithms on 2 weeks of data for each of five users. We compared against a finely tuned expert system, representing months of work, and a simple baseline that predicts the last project for the URI, last project for the resource type, or failing that, the last project.

Statistics for each of the five users’ data:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Projects</th>
<th>Time segments</th>
<th>URI subpath</th>
<th>Title words</th>
<th>Body words</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>26</td>
<td>345</td>
<td>1401</td>
<td>1058</td>
<td>904</td>
<td>176</td>
</tr>
<tr>
<td>User 2</td>
<td>20</td>
<td>308</td>
<td>1181</td>
<td>941</td>
<td>184</td>
<td>131</td>
</tr>
<tr>
<td>User 3</td>
<td>19</td>
<td>220</td>
<td>1539</td>
<td>139</td>
<td>608</td>
<td>107</td>
</tr>
<tr>
<td>User 4</td>
<td>18</td>
<td>139</td>
<td>1680</td>
<td>139</td>
<td>68</td>
<td>107</td>
</tr>
<tr>
<td>User 5</td>
<td>17</td>
<td>123</td>
<td>167</td>
<td>123</td>
<td>139</td>
<td>107</td>
</tr>
</tbody>
</table>

NB and PA were trained online. LR and SVM were trained on four users and tested on the remaining one. We report total errors to the right and several accuracies to the far right.

Niche and Per-URI Accuracy

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Niche</th>
<th>Per-URI Accuracy</th>
<th>Per-URI Accuracy 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>65.4%</td>
<td>55.4%</td>
<td>55.4%</td>
<td></td>
</tr>
<tr>
<td>PA</td>
<td>66.7%</td>
<td>55.4%</td>
<td>55.4%</td>
<td></td>
</tr>
<tr>
<td>LR</td>
<td>69.0%</td>
<td>64.7%</td>
<td>65.2%</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>90.1%</td>
<td>68.1%</td>
<td>70.1%</td>
<td></td>
</tr>
</tbody>
</table>

RESULTS

- Salience features greatly help NB and PA.
- SVM s' is more accurate than expert system for every single user.
- Even Baseline is fairly accurate, because most resources are visited several times.
- SVM accuracy increases in the second half of new URI predictions (unlike baseline and expert), suggesting long-term gains.