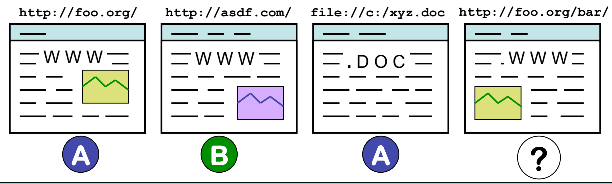
Using Salience to Segment Desktop Activity into Projects

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 compensated 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.

Assumptions
- Users specify projects
- Users correct wrong predictions quickly and reliably

Requirements:
- Prediction in < 100ms
- Adapt quickly to new projects
- Few "stupid" mistakes

TASK: Predict project label, p_n, given
Current and previous actions: $(a_0, \ldots, a_{n-1}, a_n)$
Previous project labels: $(p_0, \ldots, p_{n-1})$

APPROACH: Linear classifier
Simple, fast, effective (with the right features)

$$\underbrace{c(x)}_{\text{Predicted project}} = \arg\max_{p \in P} \underbrace{\mathbf{w}}_{\text{Weights}} \cdot \underbrace{\Psi(x,p)}_{\text{Features}}$$

FEATURES: Salience Links Content to Context

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

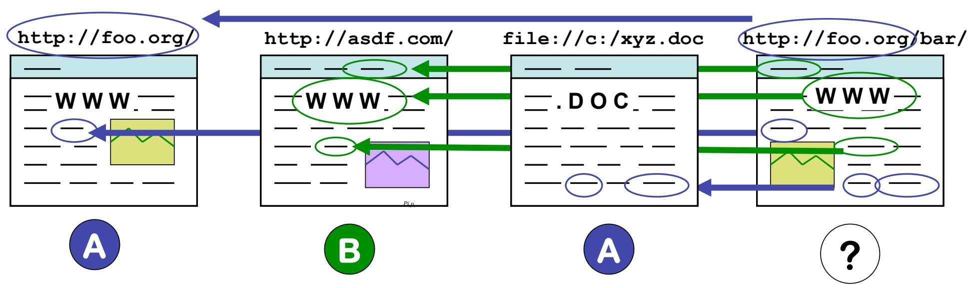
General
- Full URI
- URI subpath
- Title words
- Body words
- Type

Email Only
- Sender
- Recipients
- Thread-ID
- Subject
- Attachments
- Folder

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

Ψ(x,A) = {"URI subpath A" = 1, "Body words A" = 3}
Ψ(x,B) = {"Title words B" = 1, "Body words B" = 1, "Type B" = 1}

Example: Shared Salience Features

Ψ(x,A) = {"URI subpath" = 1, "Body words" = 3}
Ψ(x,B) = {"Title words" = 1, "Body words" = 1, "Type" = 1}

ALGORITHMS: Four Standard Approaches

Naïve Bayes (NB)

Assume observed features $\Psi(x_t)$ are independent given class label (project).

$$P(\hat{p}_t \mid x_t) = \frac{1}{Z_t} P(p_t) \prod_i P(\Psi_i(x_t) \mid \hat{p}_t)$$

Log probability is a linear model.
Weights are log conditional probabilities.

PRO: Simple and fast.
Often surprisingly effective.
CON: Overly strong assumptions.

Passive-Aggressive (PA)

After each example, update weights so that hinge loss l_t on most recent example is zero.

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t(\underbrace{\Psi(x_t,p_t)}_{\text{Features of true project}} - \underbrace{\Psi(x_t,p_t')}_{\text{Features of predicted project}})$$

$$\text{where } \tau_t = \frac{l_t}{\|\Psi(x_t,p_t) - \Psi(x_t,p_t')\|^2 + \underbrace{1/2C}_{\text{(Dampening term)}}}$$

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:

$$P(\hat{p}_t \mid x_t) = \frac{1}{Z_t} \exp\left(\mathbf{w} \cdot \Psi_t(x_t, \hat{p}_t)\right)$$

Log probability is linear model. Choose weights offline to minimize log loss of training data.

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:

$$h(\mathbf{w}) = \sum_t l_t$$

$$l_t = \max_{p' \neq p_t} \max(0, 1 + \mathbf{w} \cdot (\Psi(x_t, p') - \Psi(x_t, p_t)))$$

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.

	1	2	3	4	5
Projects	26	40	27	33	35
Time segments	3480	1441	5036	1681	465
Total resources	1021	390	1181	445	161
Emails	159	192	559	182	137
Web pages	829	149	570	199	0
Other	33	49	52	64	24

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.

Code	Features
R	Resource features
P	Past project features
S	Salience features
s	Shared salience features
s'	Shared salience features, predict last project for URI

Table 1. Errors on each user's data.

Method	Features	Total	User 1	2	3	4	5
Baseline		1215	229	161	508	192	126
NB	R	1096	267	141	402	178	108
	R+P	1002	220	139	368	168	107
	R+S	941	184	143	328	170	116
	R+S+P	923	176	142	323	167	115
PA	R	914	187	131	321	168	107
	R+P	899	171	127	321	169	111
	R+S	1091	216	157	399	192	127
	R+S+P	1090	217	155	397	194	127
	s	1116	214	123	494	178	107
	s'	910	172	130	348	159	**101**
	R+s'	885	167	128	329	159	102
	R+P+s'	882	166	128	327	158	103
LR	s	1047	155	121	508	158	105
	s'	871	155	121	332	158	105
SVM	s	1011	**142**	**115**	501	**149**	104
	s'	**815**	**142**	**115**	**305**	**149**	104
Expert		910	149	147	335	167	112

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.

	Per-Action Accuracy	Per-URI Accuracy	Per-URI Accuracy 2
Baseline	86.7%	55.6%	55.4%
Expert	89.0%	64.7%	65.2%
SVM s'	90.1%	68.1%	70.1%