Vowpal Wabbit 2015

Kai-Wei Chang, Markus Cozowicz, Hal Daume,
Luong Hoang, TK Huang, John Langford

http://hunch.net/~vw/

git clone
git://github.com/JohnLangford/vowpal_wabbit.git

Why does Vowpal Wabbit exist?

Why does Vowpal Wabbit exist?

1. Prove research.

Why does Vowpal Wabbit exist?

1. Prove research.
2. Curiosity.
3. Perfectionist.
4. Solve problem better.

A user base becomes addictive

1. Mailing list of >400

A user base becomes addictive

1. Mailing list of >400
2. The official strawman for large scale logistic regression @ NIPS :-)

A user base becomes addictive

1. Mailing list of >400
2. The official strawman for large scale logistic regression @ NIPS :-)
3.

An example

`wget http://hunch.net/~jl/VW_raw.tar.gz`

`vw -c rcv1.train.raw.txt -b 22 --ngram 2 --skips 4 -l 0.25 --binary` provides stellar performance in 12 seconds.

Surface details

1. BSD license, automated test suite, github repository.

Surface details

1. BSD license, automated test suite, github repository.
2. VW supports all I/O modes: executable, library, port, daemon, service (see next).

Surface details

1. BSD license, automated test suite, github repository.
2. VW supports all I/O modes: executable, library, port, daemon, service (see next).
3. VW has a reasonable++ input format: sparse, dense, namespaces, etc...

Surface details

1. BSD license, automated test suite, github repository.
2. VW supports all I/O modes: executable, library, port, daemon, service (see next).
3. VW has a reasonable++ input format: sparse, dense, namespaces, etc...
4. Mostly C++, but bindings in other languages of varying maturity (python, C#, Java good).

Surface details

1. BSD license, automated test suite, github repository.
2. VW supports all I/O modes: executable, library, port, daemon, service (see next).
3. VW has a reasonable++ input format: sparse, dense, namespaces, etc...
4. Mostly C++, but bindings in other languages of varying maturity (python, C#, Java good).
5. A substantial user base + developer base. Thanks to many who have helped.

What does Vowpal Wabbit do well?

Older:

1. Online learning. Support for real online learning.
2. Parallelization. Via allreduce
3. Scalable solutions. Logarithmic time prediction!

What does Vowpal Wabbit do well?

Older:

1. Online learning. Support for real online learning.
2. Parallelization. Via allreduce
3. Scalable solutions. Logarithmic time prediction!

Newer:

1. Problem Framing.

What does Vowpal Wabbit do well?

Older:

1. Online learning. Support for real online learning.
2. Parallelization. Via allreduce
3. Scalable solutions. Logarithmic time prediction!

Newer:

1. Problem Framing.

What does Vowpal Wabbit do well?

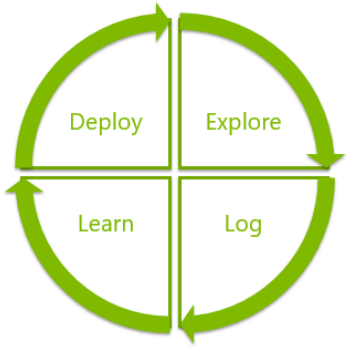
Older:

1. Online learning. Support for real online learning.
2. Parallelization. Via allreduce
3. Scalable solutions. Logarithmic time prediction!

Newer:

1. Problem Framing.

2. Learning lifecycle.



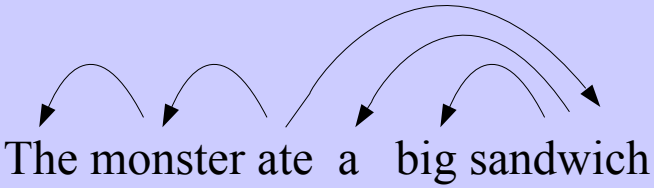
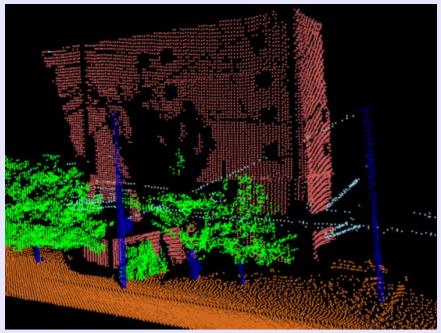
What does VW not do well?

1. GPU training.
2. Representational flexibility.

Next

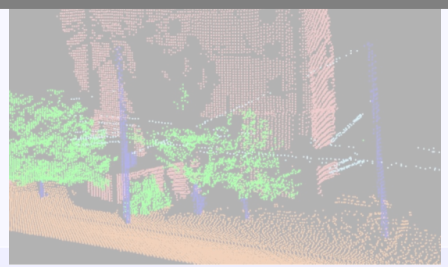
1. Learning to Search (Hal/John/Kai-Wei)
2. Active Learning (TK)
3. System Integration (Markus)
4. Client side Decision Service (Luong)

What are joint predictions?

Task	Input	Output
Machine Translation	Ces deux principes se tiennent à la croisée de la philosophie, de la politique, de l'économie, de la sociologie et du droit.	Both principles lie at the crossroads of philosophy, politics, economics, sociology, and law.
Sequence Labeling	The monster ate a big sandwich	Det Noun VerbDetAdj Noun The monster ate a big sandwich
Syntactic Analysis	The monster ate a big sandwich	
3d point cloud classification	3d range scan data	
...many more...		

What are joint predictions?

Task	Input	Output
Machine Tr	Ces deux principes se tiennent à	Both principles lie at the
Se La		
Sy An		h
3d classification		
...many more...		



Structured Prediction Haiku

A joint prediction
Across a single input
Loss measured jointly

We want to minimize...

- **Programming complexity.** Most joint prediction problems are *not* addressed using structured learning because of programming complexity.
- **Test loss.** If it doesn't work, game over.
- **Training time.** Debug/develop productivity, hyperparameter tuning, maximum data input.
- **Test time.** Application efficiency.

Programming complexity

search_sequencetask.cc

File Edit Options Buffers Tools C++ YASnippet Development Cscope Help

```
namespace SequenceTask {
  void initialize(Search::search& sch, size_t& num_actions, po::variables_map& vm) {
    sch.set_options( Search::AUTO_CONDITION_FEATURES  |
                     Search::AUTO_HAMMING_LOSS        |
                     Search::EXAMPLES_DONT_CHANGE     |
                     0);
  }

  void run(Search::search& sch, vector<example*>& ec) {
    for (int i=0; i<ec.size(); i++) {
      action oracle     = MULTICLASS::get_example_label(ec[i]);
      size_t prediction = Search::predictor(sch, i+1).set_input(*ec[i]).set_oracle(oracle)
          .set_condition_range(i, sch.get_history_length(), 'p').predict();

      if (sch.output().good())
        sch.output() << prediction << ' ';
    }
  }
}
```

-:**- search_sequencetask.cc 6% (34,0) Git-master (C++/l BufFace AC yas Abbrev)

Python interface to VW

Library interface to VW (*not* a command line wrapper)

It is *actually* documented!!!

Allows you to write code like:

```
import pyvw
vw = pyvw.vw("--quiet")
ex1 = vw.example("1 |x a b |y c")
ex2 = vw.example({'x': ['a', ('b', 1.0)], \
                  'y': ['c']})
ex1.learn()
print ex2.predict()
```

iPython Notebook for Learning to Search

IP[y]: Notebook Learning_to_Search Last Checkpoint: Oct 03 14:43 (autosaved)

File Edit View Insert Cell Kernel Help

Markdown Cell Toolbar: None

The _run function executes the sequence of decisions on a given input. The input will be of whatever type our data is (so, in the above example, it will be a list of (label,word) pairs).

Here is a basic implementation of sequence labeling:

In [39]:

```
class SequenceLabeler(pyvw.SearchTask):
    def __init__(self, vw, sch, num_actions):
        pyvw.SearchTask.__init__(self, vw, sch, num_actions)
        sch.set_options( sch.AUTO_HAMMING_LOSS | sch.AUTO_CONDITION_FEATURES )

    def _run(self, sentence):
        output = []
        for n in range(len(sentence)):
            pos,word = sentence[n]
            with self.vw.example({'w': [word]}) as ex:
                pred = self.sch.predict(examples=ex, my_tag=n+1, oracle=pos, condition=(n,'p'))
                output.append(pred)
        return output
```

http://tinyurl.com/pyvwsearch
http://tinyurl.com/pyvwtalk
http://tinyurl.com/lolstalk2

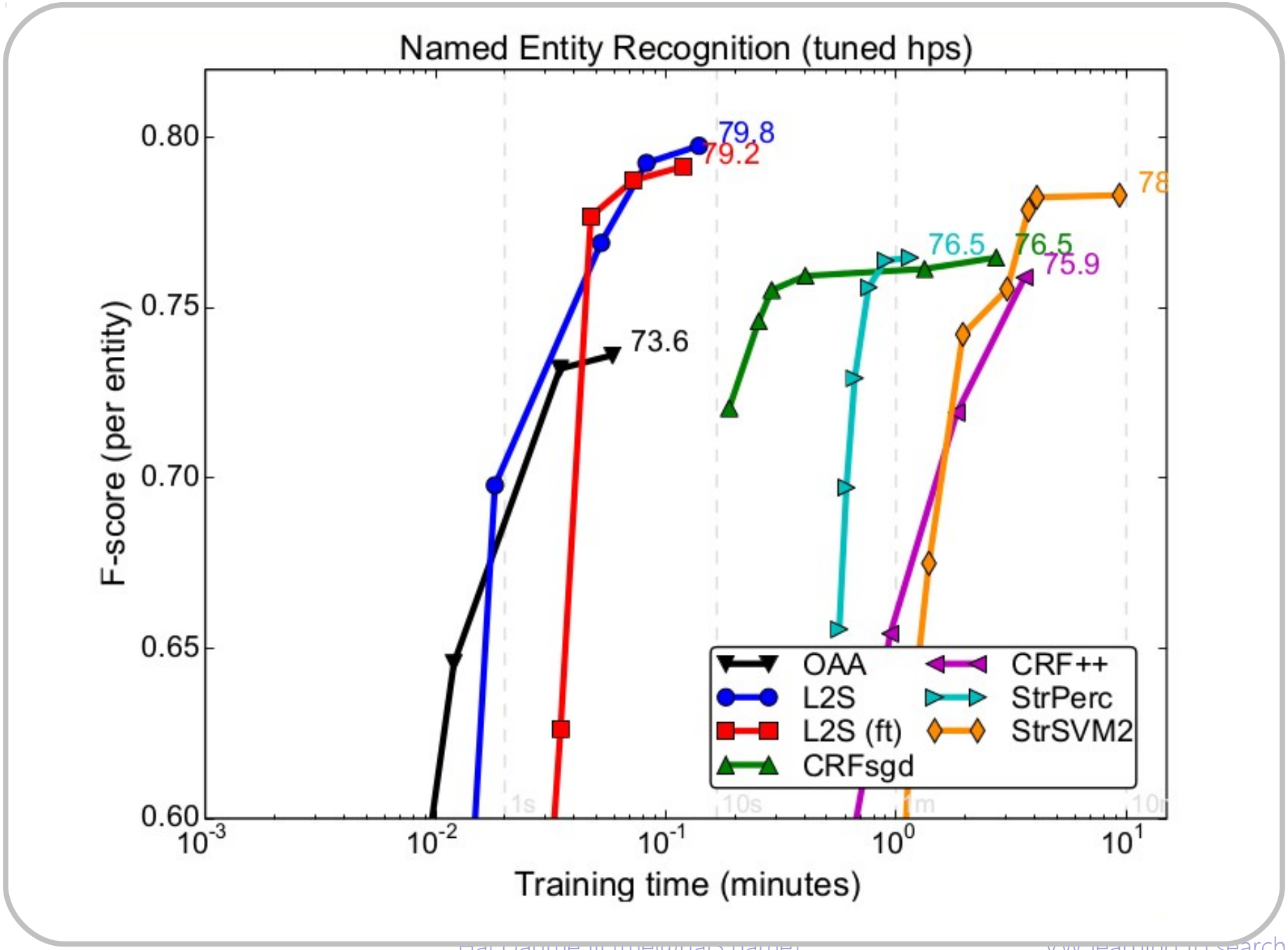
State of the art accuracy in....

- Part of speech tagging (1 million words)
  - vw: 6 lines of code 10 seconds to train
  - CRFsgd: 1068 lines 6 minutes
  - CRF++: 777 lines hours
- Named entity recognition (200 thousand words)
  - vw: 30 lines of code 5 seconds to train
  - CRFsgd: 1 minute (subopt accuracy)
  - CRF++: 10 minutes (subopt accuracy)
  - SVM^{str}: 876 lines 30 minutes (subopt accuracy)

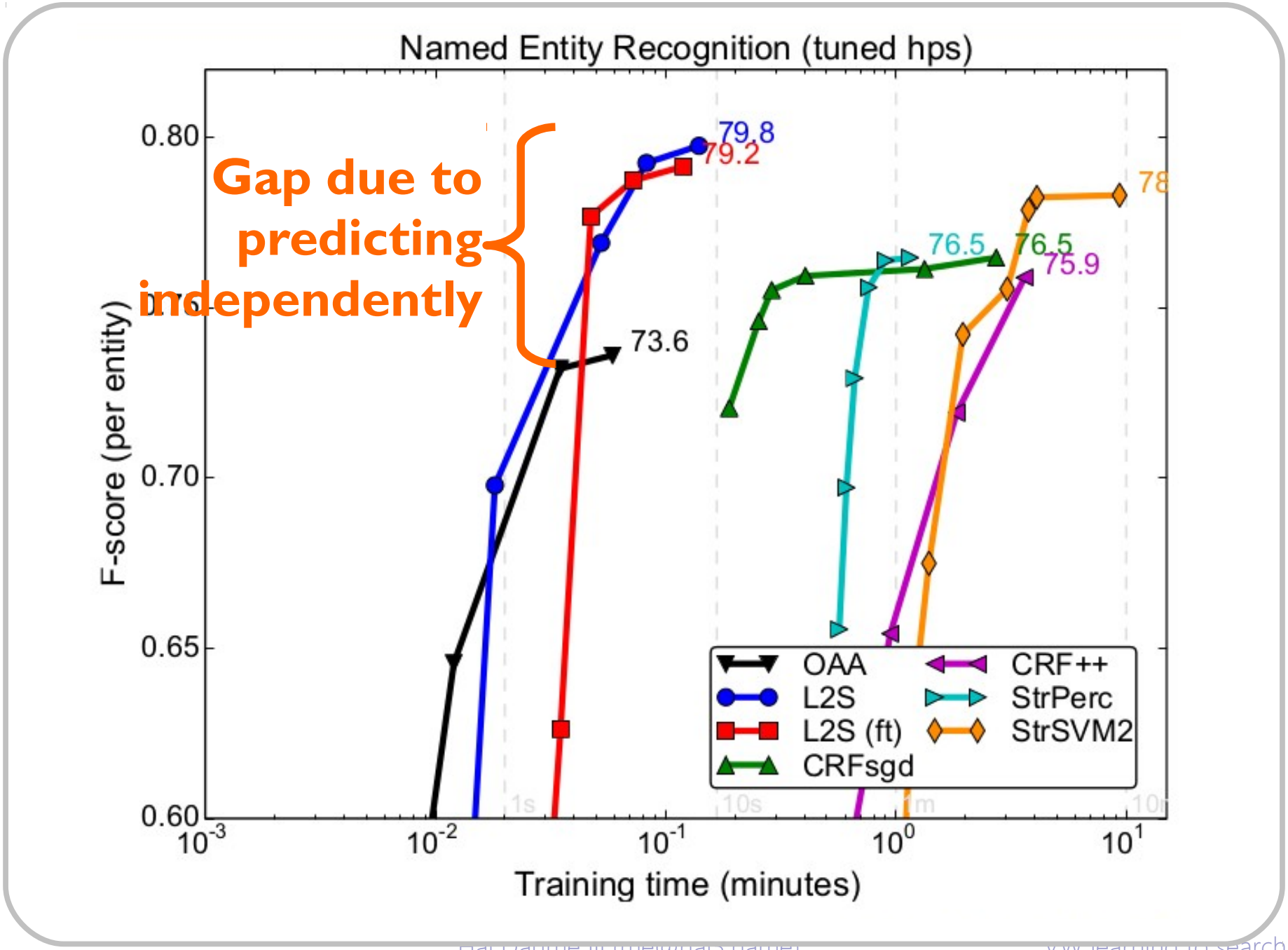
State of the art accuracy in....

- Part of speech tagging (1 million words)
  - wc: 3.2 seconds
  - vw: 6 lines of code 10 seconds to train
  - CRFsgd: 1068 lines 6 minutes
  - CRF++: 777 lines hours
- Named entity recognition (200 thousand words)
  - wc: 0.8 seconds
  - vw: 30 lines of code 5 seconds to train
  - CRFsgd: 1 minute (subopt accuracy)
  - CRF++: 10 minutes (subopt accuracy)
  - SVM^{str}: 876 lines 30 minutes (subopt accuracy)

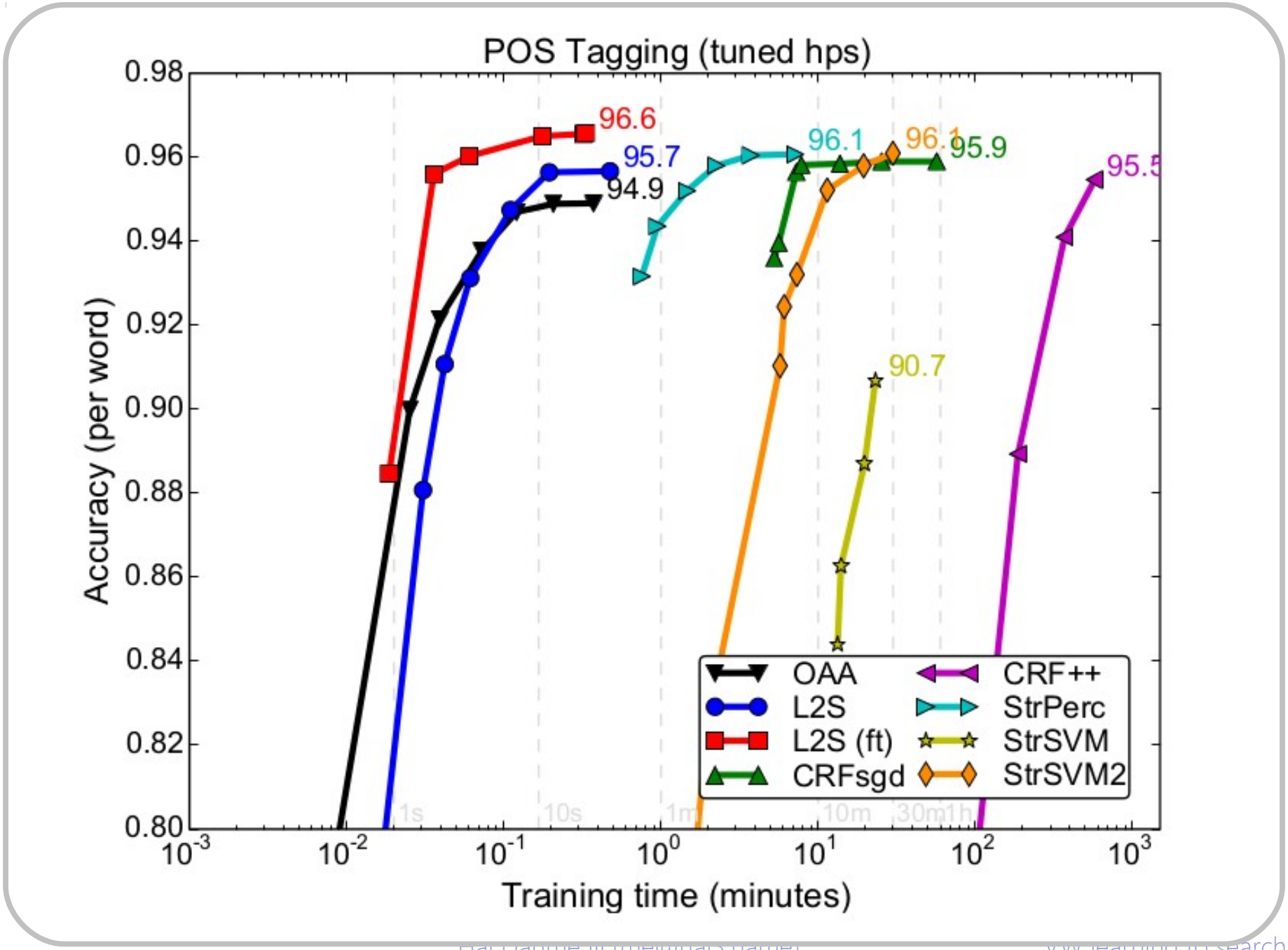
Training time versus test accuracy



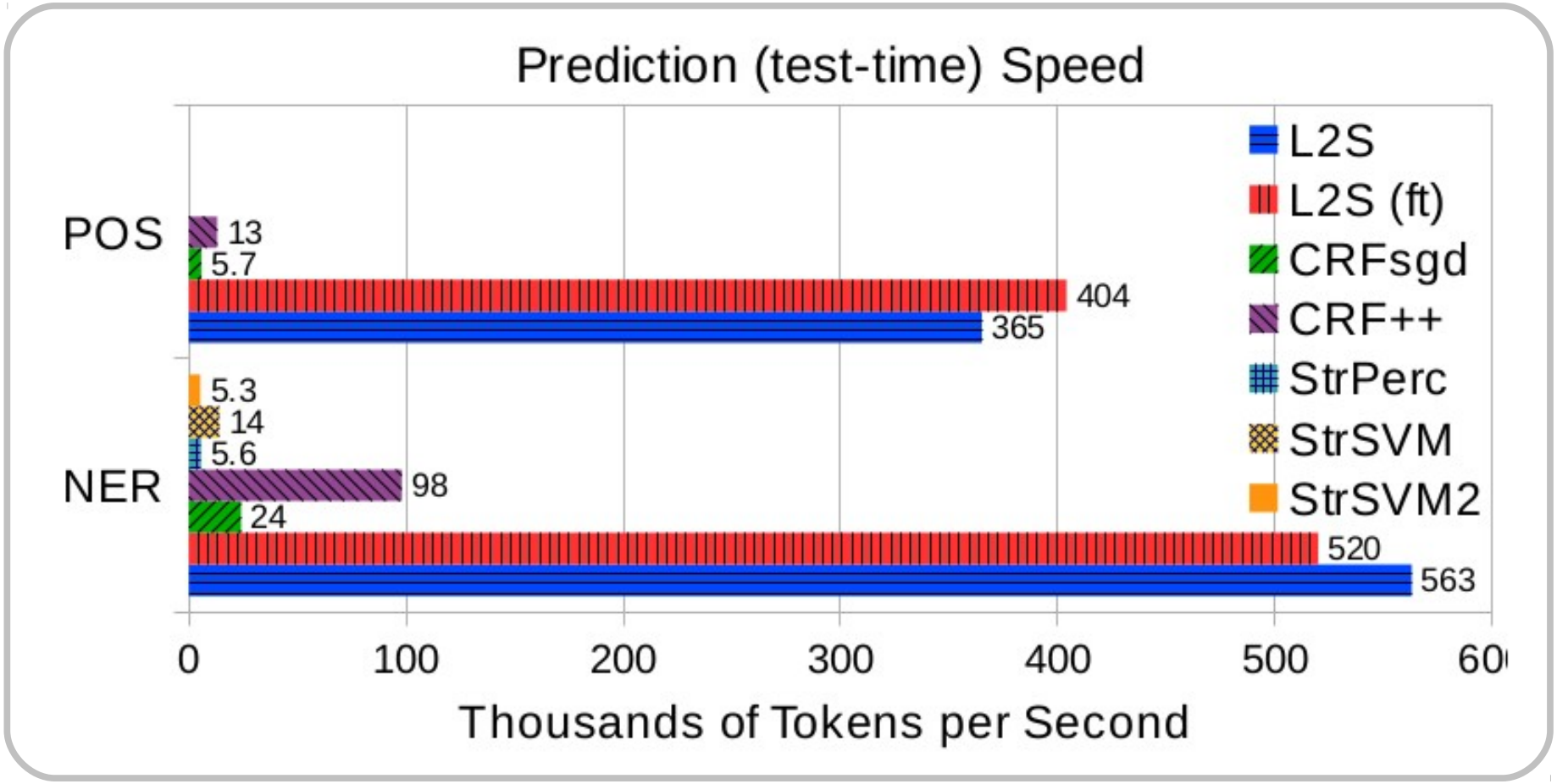
Training time versus test accuracy



Training time versus test accuracy



Test time speed

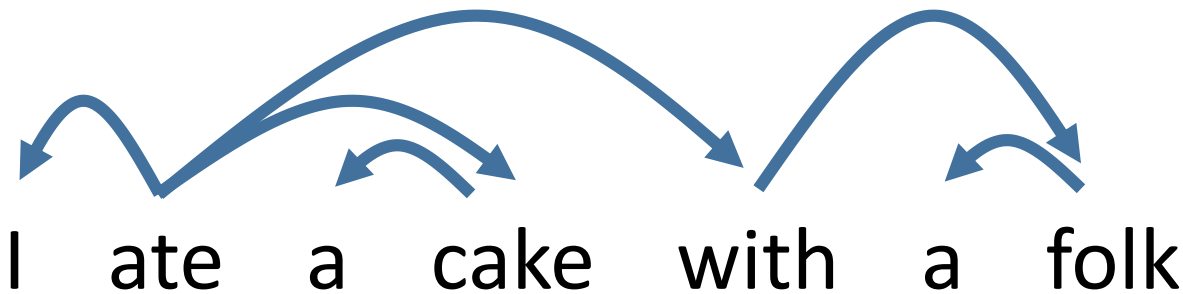


Possibly the fastest test-time prediction out there, and without "label dictionary" hacks

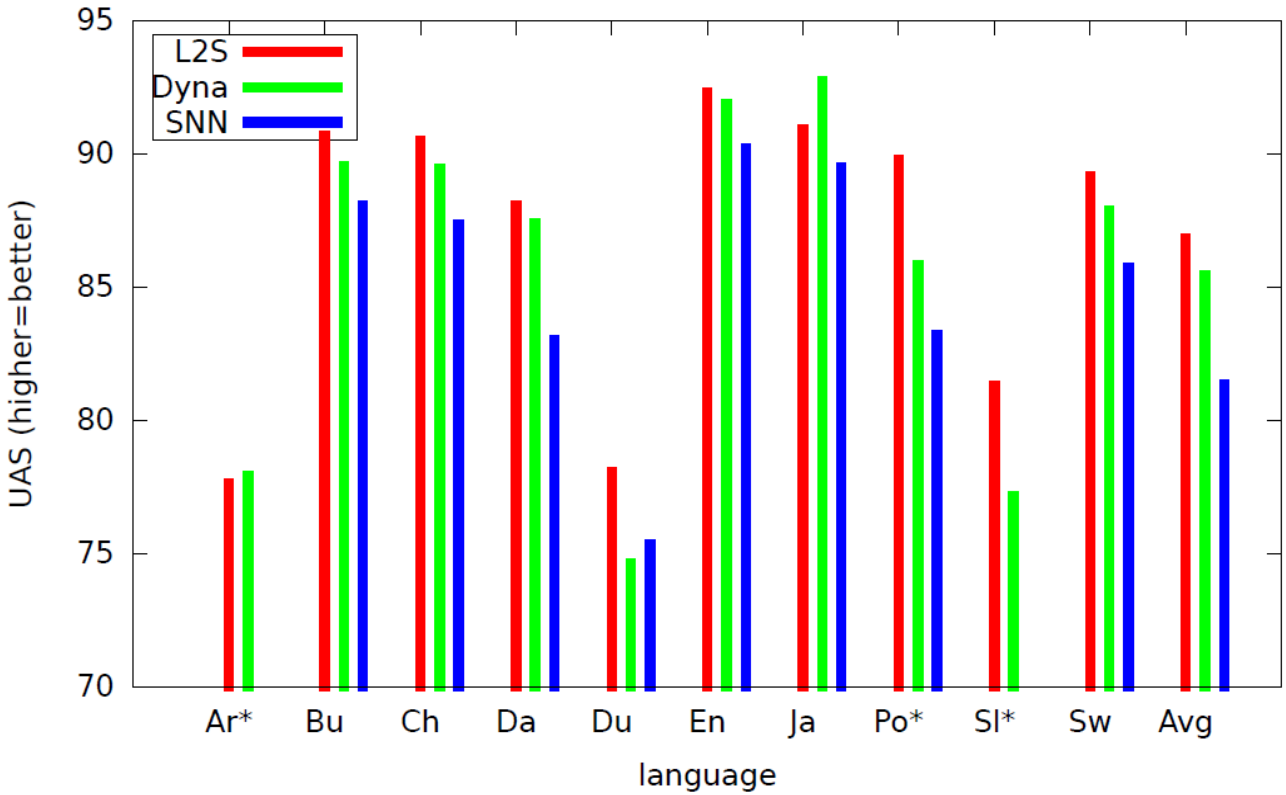
Command-line usage

```
% wget http://bilbo.cs.uiuc.edu/~kchang10/tmp/wsj.vw.zip

% unzip wsj.vw.zip

% vw -b 24 -d wsj.train.vw -c --search_task sequence \
   --search 45 --search_neighbor_features -1:w,1:w   \
   --affix -1w,+1w -f wsj.weights

<chat with your neighbor for 10 seconds>

% vw -t -i wsj.weights wsj.test.vw

<wait 0.15 seconds for 96.4% accuracy>
```

Identifying Relationship between Words



Dependency Parser in VW



- # lines of code ~ 300

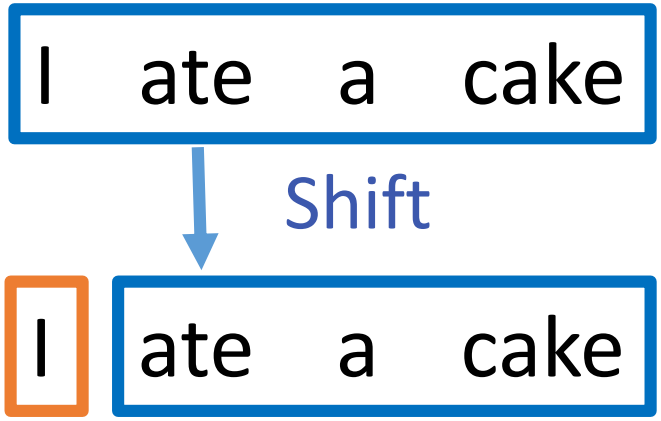
[Arxiv 15a]: Learning to search dependencies

Shift-Reduce Parser

- Maintain a buffer and a stack
- Make predictions from left to right
- Three types of actions:
  Shift, Reduce-Left, Reduce-Right

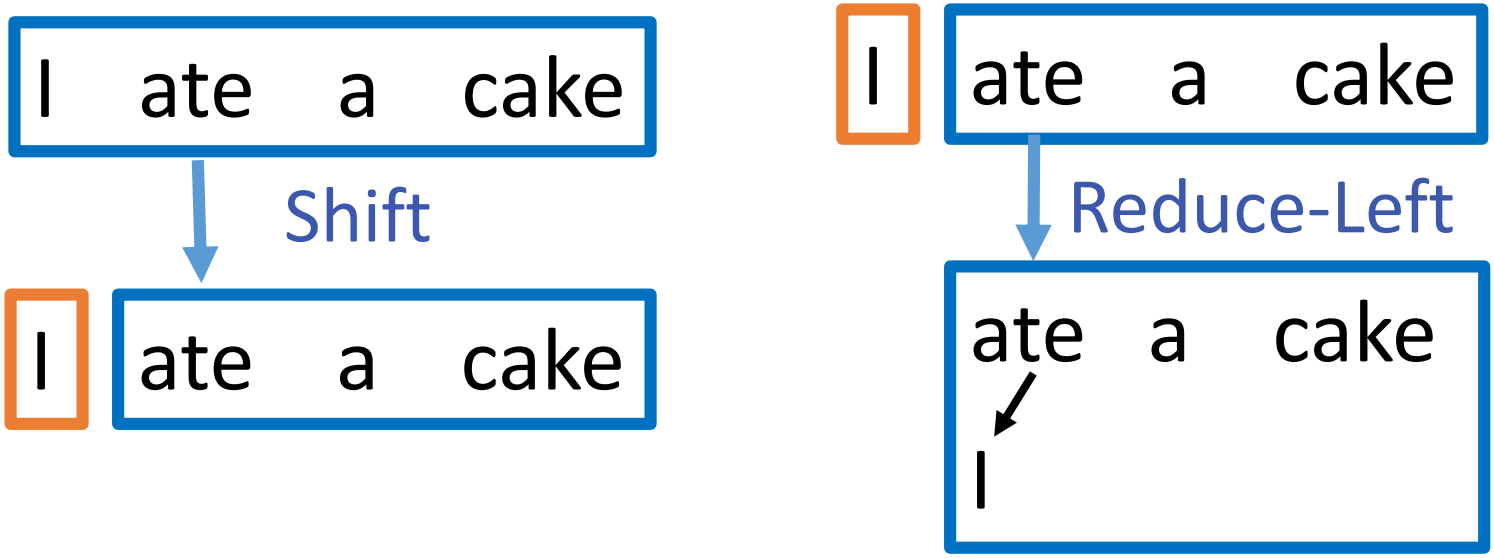
Shift-Reduce Parser

- Maintain a buffer and a stack
- Make predictions from left to right
- Three types of actions:
  Shift, Reduce-Left, Reduce-Right



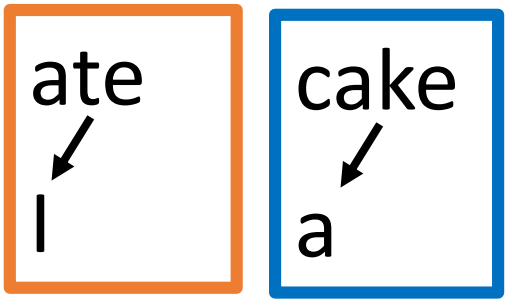
Shift-Reduce Parser

- Maintain a buffer and a stack
- Make predictions from left to right
- Three types of actions:
  Shift, Reduce-Left, Reduce-Right



Features

- Lexicon & POS tags of …
  - top three words in the stack,
  - first three words in the buffer,
  - and their children
- Combination (quadratic, cubic) of features



RunParser(*sentence*)

```
stack S ← {Root}
buffer B ← [words in sentence]
arcs A ← ∅
while  B ≠ ∅ or |S| > 1 do
    ValidActs ← GetValidActions(S, B)
    features ← GetFeat(S, B, A)
    ref ← GetGoldAction(S, B)
    action ← predict(features, ref, ValidActs)
    S, B, A ← Transition(S, B, A, action)
end while
loss(A[w] ≠ A*[w], ∀w ∈ sentence)
return output
```

Run the Parser

- Under demo/dependencyparsing
- Data:

```
2 2 2:nmod|w ms. |p nnp
3 5 3:sub|w haag |p nnp
0 8 0:root|w plays |p vbz
3 7 3:obj|w piano|p nn
3 4 3:p|w . |p .
```

Ms. Haag plays piano .

Active Learning in VW

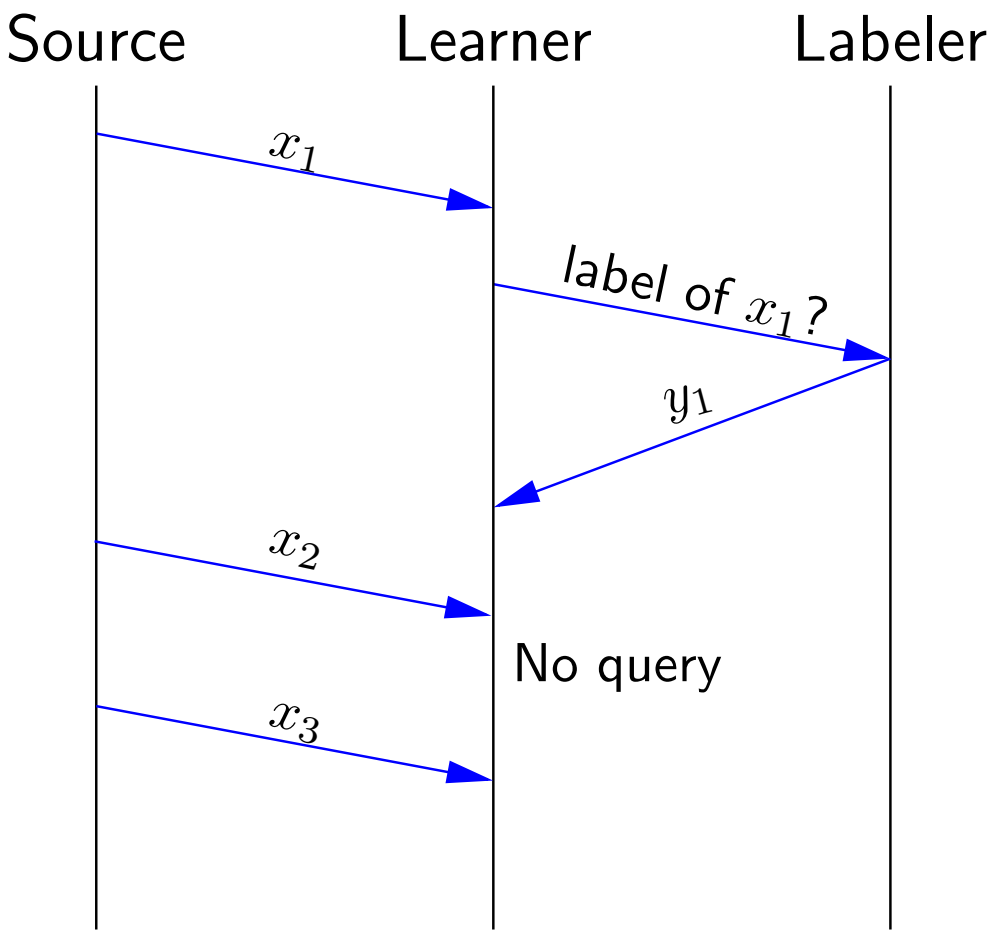
Streaming Selective Sampling

Repeat:

1. Receive a new $x \overset{i.i.d.}{\sim} \mathcal{D}_{\mathcal{X}}$.
2. Query for label? Yes/no
3. If yes, obtain label y.

Goal: Maximize classifier accuracy per label query

Key step: query decision



Active Learning in VW: Simulation Mode

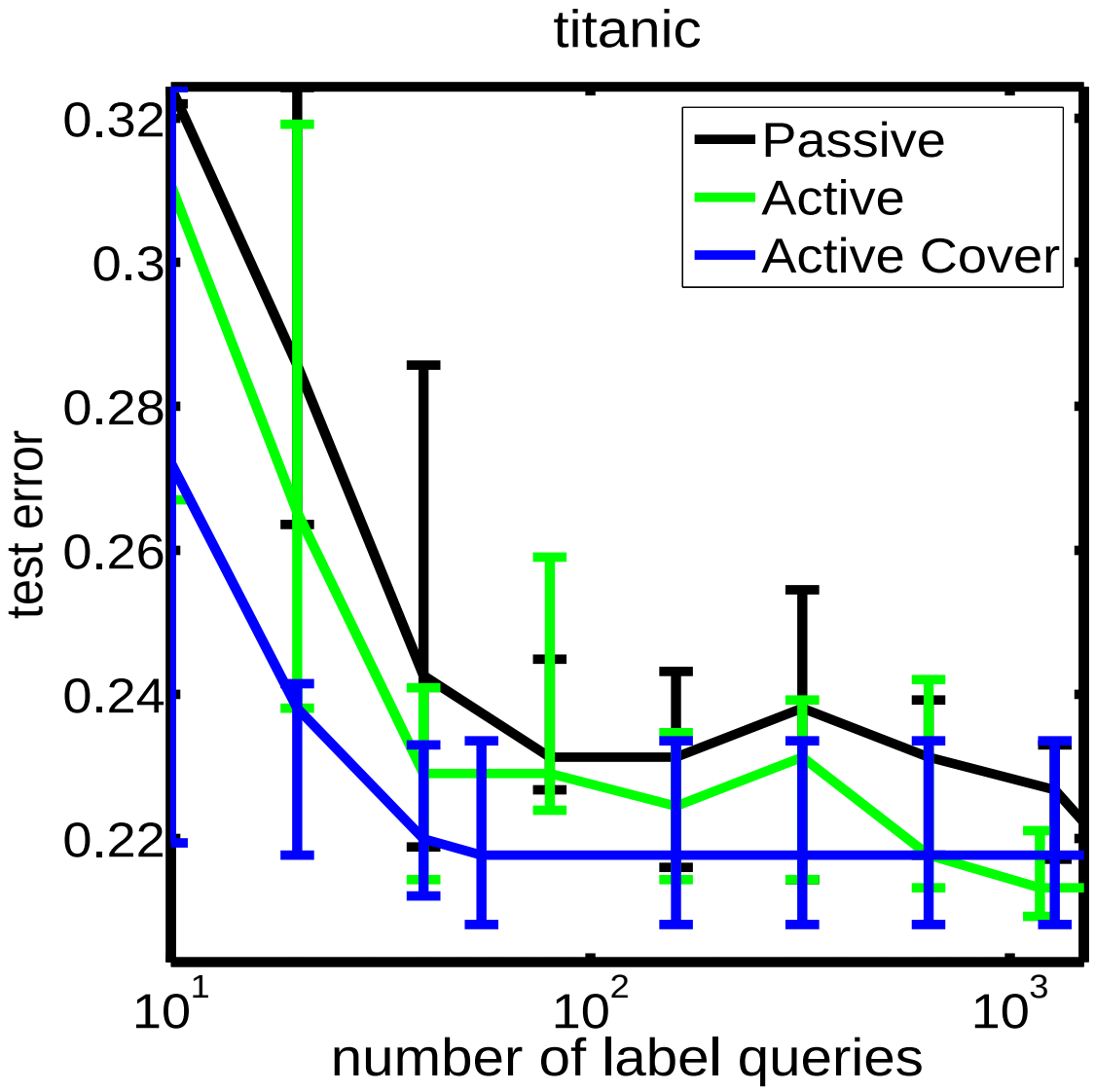
```
vw --binary --active --simulation --mellowness 0.01
labeled.data
```

`--mellowness`: small value leads to few label queries

```
vw --binary --active --cover 10 --mellowness 0.01
train.data
```

`--cover`: number of classifiers used to measure uncertainty about the label. Use a large `-b` (e.g. 29) with a large `--cover` (e.g. 50).

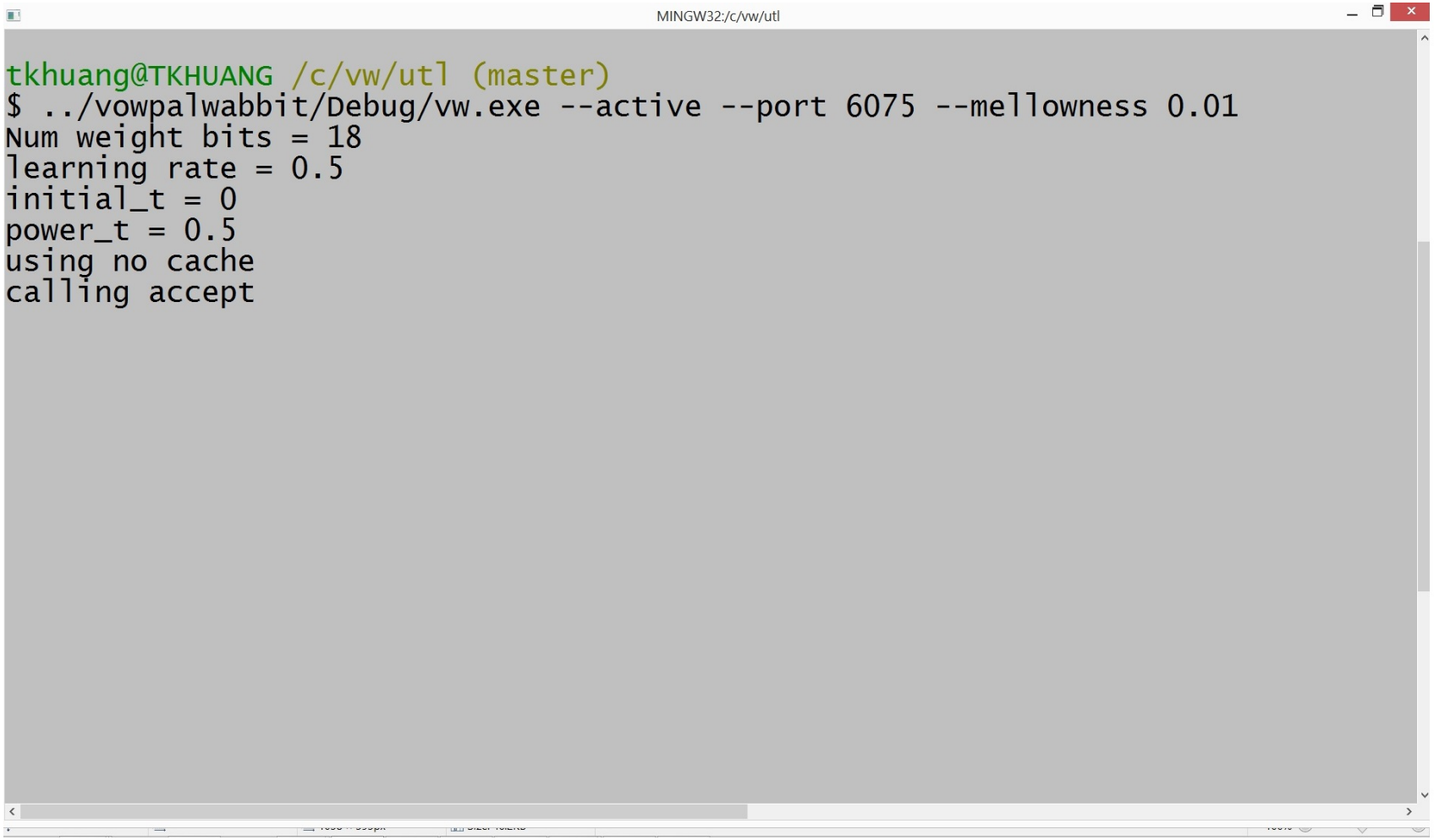
Active Learning in VW: Simulation Mode



Active Learning in VW: Interactive Mode

```
vw --active --port 6075 --mellowness 0.01
```

`--port`: port number VW is listening



Active Learning in VW: Interactive Mode

```
python utl/active_interactor.py -v -m -o labeled.dat
localhost 6075 unlabeled.dat
```

MINGW32:/c/vw/utl

```
tkhuang@TKHUANG /c/vw/utl (master)
$ /c/Python27/python.exe active_interactor.py -v -m -o labeled.da
t localhost 6075 ./unlabeled.dat
connecting to localhost:6075 ...
done
sending unlabeled examples ...
sending unlabeled '|f crew adult male\n'

request for example 0: tag="", prediction=0: |f crew adult male

Provide? [0/1/skip]: 0
sending unlabeled '|f 1st adult male\n'

request for example 1: tag="", prediction=-0.473623: |f 1st adult male

Provide? [0/1/skip]: 1
sending unlabeled '|f crew adult male\n'

request for example 2: tag="", prediction=0.365977: |f crew adult male

Provide? [0/1/skip]: 0
```

New C# API

- Significant performance improvement for data transfer
- Intuitive memory management through IDisposable
- Binary available on nuget.org

New C# API | string data

```
using (var vw = new VowpalWabbit("--quiet"))
{
      vw.Learn("1 |f 13:3.9 24:3.4 69:4.6");
      var prediction = vw.Predict(
            "|f 13:3.9 24:3.4 69:4.6",
            VowpalWabbitPredictionType.Scalar);
      vw.SaveModel("output.model");
}
```

New C# API | object data

```
public class MyExample
{
      [Feature(FeatureGroup = 'p')]
      public float Income { get; set; }

      [Feature(Enumerize = true)]
      public int Age { get; set; }
}

new MyExample { Income = 40, Age = 25 }
→  "|p Income:40.0 | Age25"
```

New C# API | object data

```
using (var vw = new VowpalWabbit<MyExample>(""))
{
    var ex = new MyExample { Income = 40, Age = 25 };
    var label = new SimpleLabel { Label = 1 };

    vw.Learn(ex, label);
    var prediction = vw.Predict(ex,
        VowpalWabbitPredictionType.Scalar);
}
```

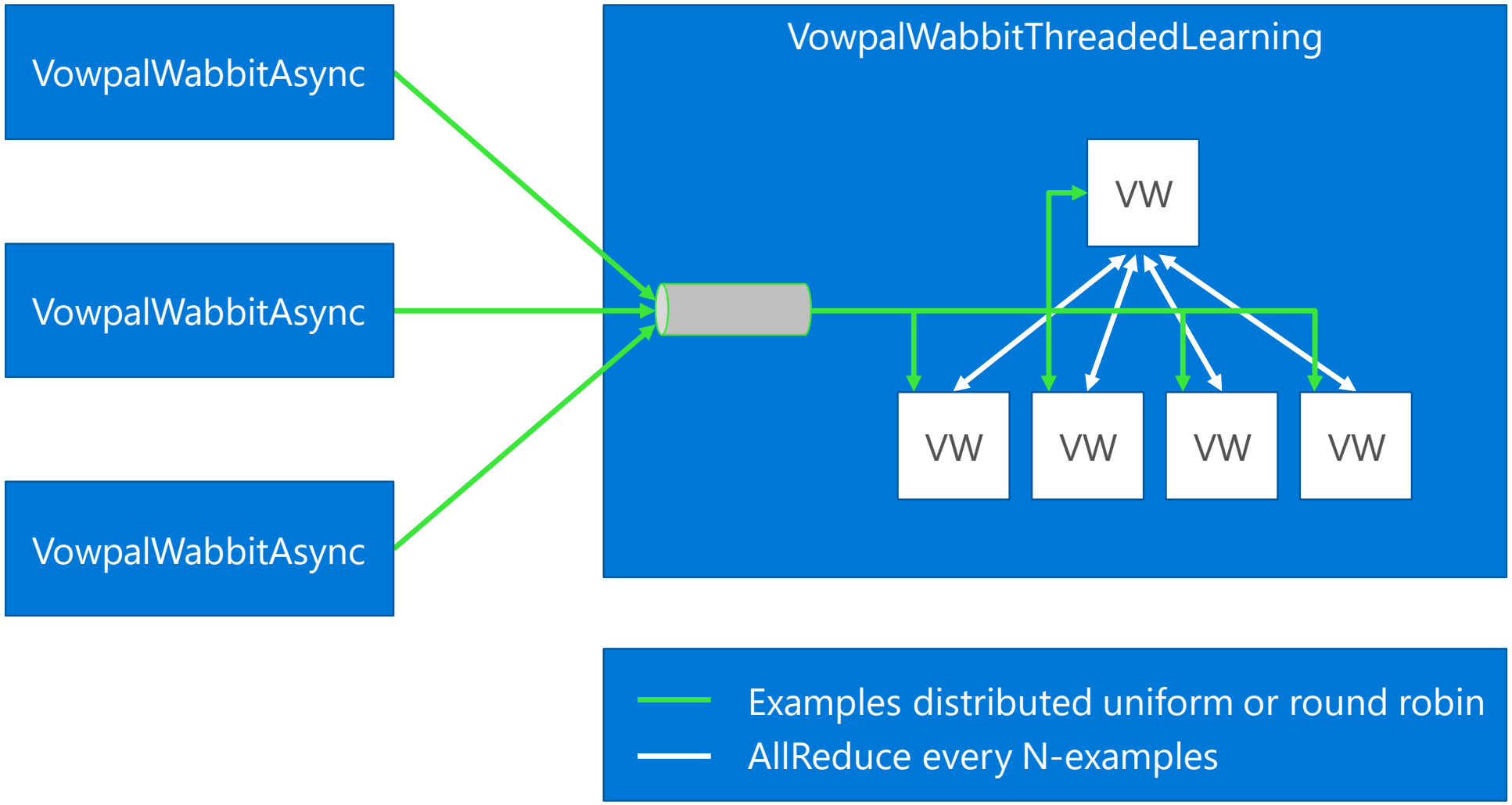
Multi-threaded | prediction

- Common use case to score multi-threaded
- Vowpal Wabbit is **not** thread-safe
- New C/C++ API to share model
- Instance pooling done in language binding

Multi-threaded | prediction

```
var vwModel = new VowpalWabbitModel("-t -i m1.model");
using (var pool = new VowpalWabbitThreadedPrediction<MyExample>(vwModel))
{
      // thread-safe
      using (var vw = pool.GetOrCreate())
      {
            // vw.Value is not thread-safe
            vw.Value.Predict(example);
      }

      // thread-safe
      pool.UpdateModel(new VowpalWabbitModel("-t -i m2.model"));
}
```

Multi-threaded | learning



Multi-threaded | learning

```
var settings = new VowpalWabbitSettings(
  parallelOptions: new ParallelOptions { MaxDegreeOfParallelism = 16 },
  exampleCountPerRun: 2000);

using (var vw = new VowpalWabbitThreadedLearning(settings))
{
  using (var vwFeeder = vw.Create<MyExample>())
  {
    var prediction = await vwFeeder.Learn(example, label,
        VowpalWabbitPredictionType.Scalar);
  }

  await vw.Complete();
}
```

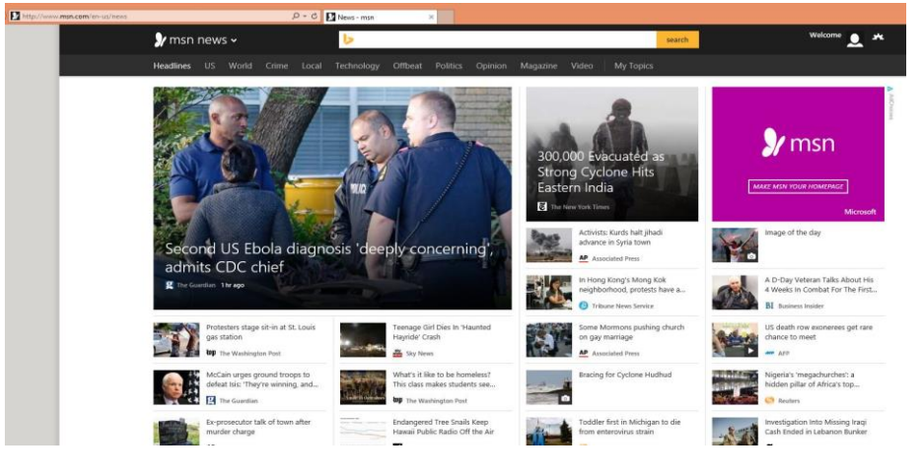
Multiworld-Testing as a Service (MTaaS)

Luong (Louie) Hoang

MWT Team @ Microsoft Research

(Partial slide credit to Alekh Aargawal)

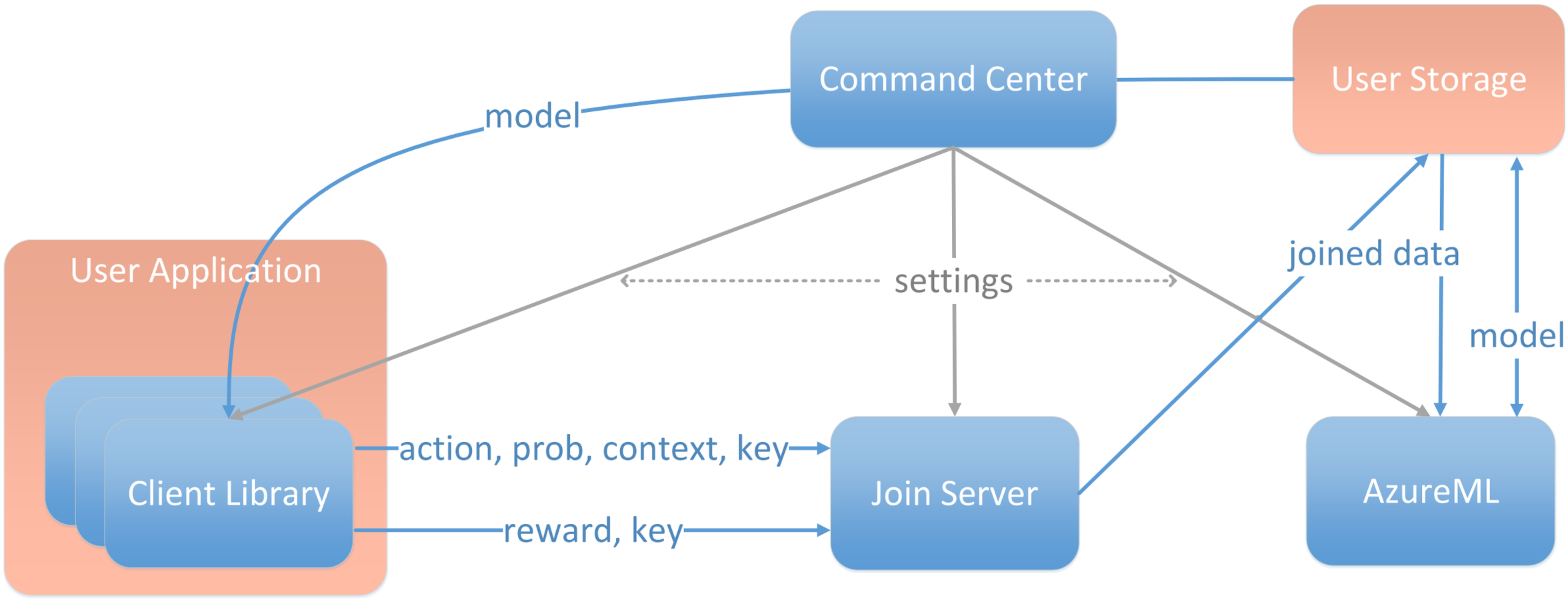
The Problem



- Loop:
  - 1. User **arrives** at MSN with browsing history, user account, previous visits,…
  - 2. Microsoft **chooses** news stories, ads, …
  - 3. User **responds** to content (clicks/navigation,…)

- Goal: Choose content to yield desired user behavior

The Service


Command Center
User Storage
model
User Application
settings
joined data
model
Client Library
action, prob, context, key
Join Server
AzureML
reward, key

The Code

```
. . .
var serviceConfig = new DecisionServiceConfiguration<UserContext>
(
    authorizationToken: MwtServiceToken,
    explorer: new EpsilonGreedyExplorer<UserContext>(. . .)
);
var service = new DecisionService<UserContext>(serviceConfig);
uint topicId = service.ChooseAction(uniqueKey: userId, context: userContext);
. . .
```

Where?

- aka.ms/mwt
- github.com/multiworldtesting

Further Pointers

Learning to Search tutorial:
`http://hunch.net/~l2s`

Talks on Decision Service this afternoon.
2:30 @Learning Systems
4:30 @Adaptive Learning More details:

`http://aka.ms/mwt` Mailing list:

`vowpal_wabbit@yahoogroups.com`