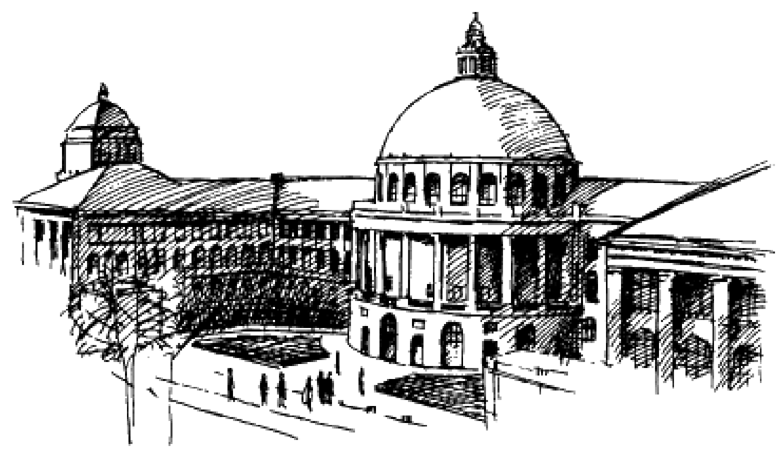
Machine Learning for Code Analytics

Martin Vechev Veselin Raychev

Department of Computer Science
ETH Zurich

ETH Zürich



Machine Learning for Programming

Probabilistically likely solutions to problems impossible to solve otherwise

Publications:

- Predicting Program Properties from "Big Code", ACM POPL'15
- Programming with Big Code, SNAPL 2015
- Code Completion with Statistical Language Models, ACM PLDI'14
- Machine Translation for Programming Languages, ACM Onward'14
- Bug Localization with Statistical Language Models, ETH TR
- Fast and Precise Statistical Code Completion, ETH TR

Tools:

JSNICE **(used worldwide)**

statistical de-obfuscation

SLANG

statistical code synthesis

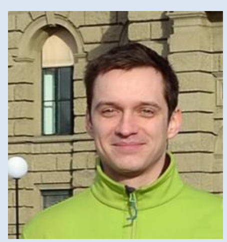
SAGE

statistical feedback generation

Martin Vechev

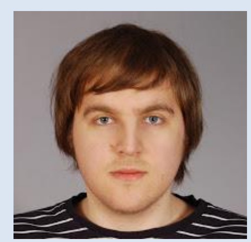
Andreas Krause

Veselin Raychev



Pavol Bielik

Christine Zeller



Svetoslav Karaivanov

Pascal Roos

Benjamin Bischel

More information: http://www.srl.inf.ethz.ch/

ETH Zürich

Tutorial Outline

- Motivation
  - Potential applications
- Statistical language models
  - N-gram and Recurrent Networks, Smoothing
  - Application: code completion
- Graphical Models
  - Markov Networks, Conditional Random Fields
  - Inference in Markov Networks
  - Learning in Markov Networks
  - Application: predicting names and types
- Hands-on session with Nice2Predict

Machine Learning for Programming

Applications

Intermediate Representation

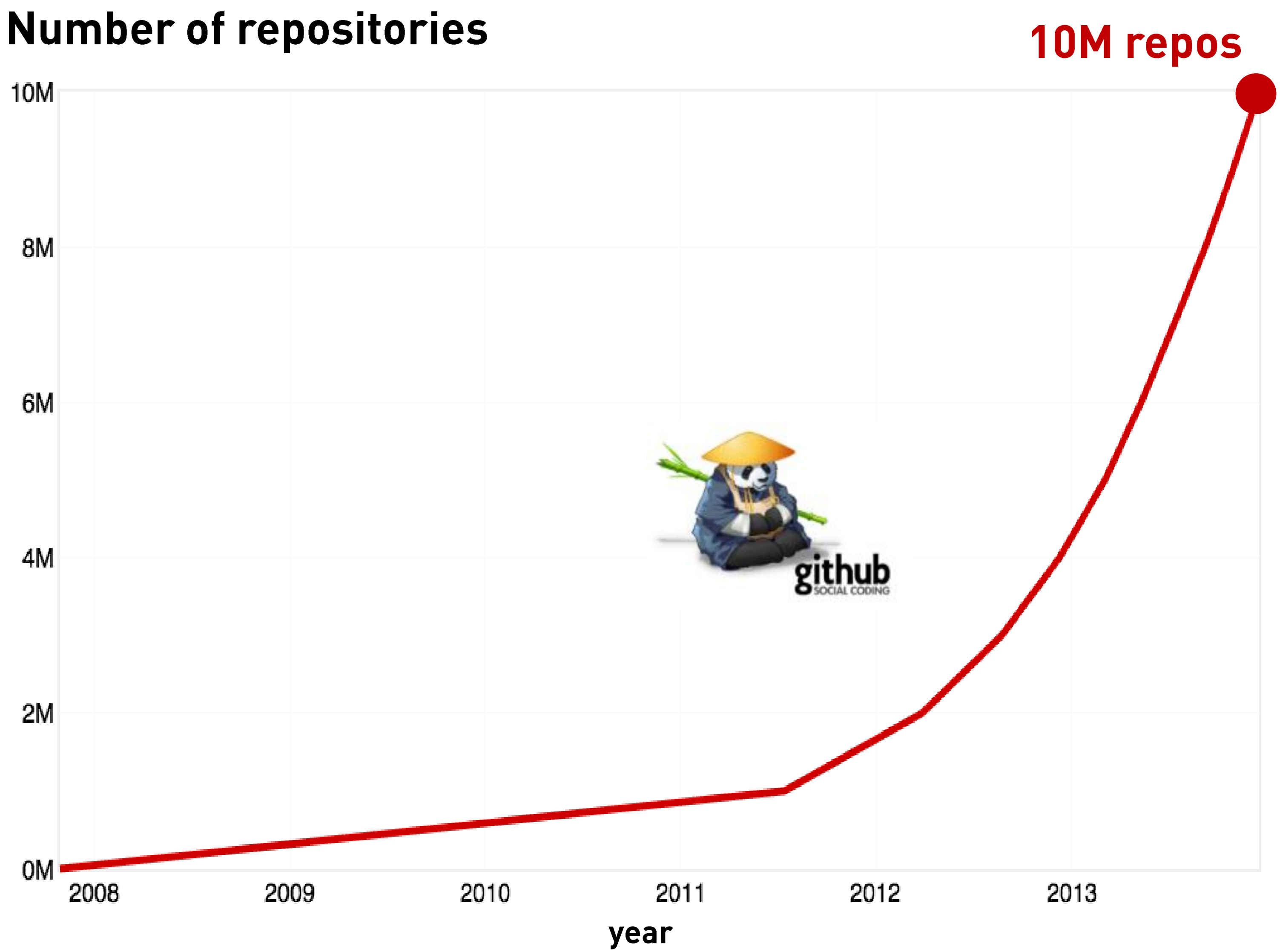
Analyze Program (PL)

Train Model (ML)

Query Model (ML)

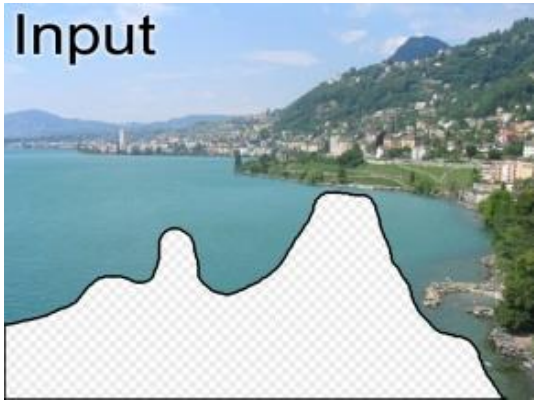
Machine Learning for Programming

Applications	Code completion, Deobfuscation, Program synthesis, Feedback generation, Translation
Intermediate Representation	Sequences (sentences), Trees, Translation Table, Graphical Models (CRFs), Feature Vectors
Analyze Program (PL)	typestate analysis, scope analysis, control-flow analysis, alias analysis
Train Model (ML)	Neural Networks, N-gram language model, SVM, Structured SVM
Query Model (ML)	argmax P(y | x), $y \in \Omega$; Greedy MAP inference

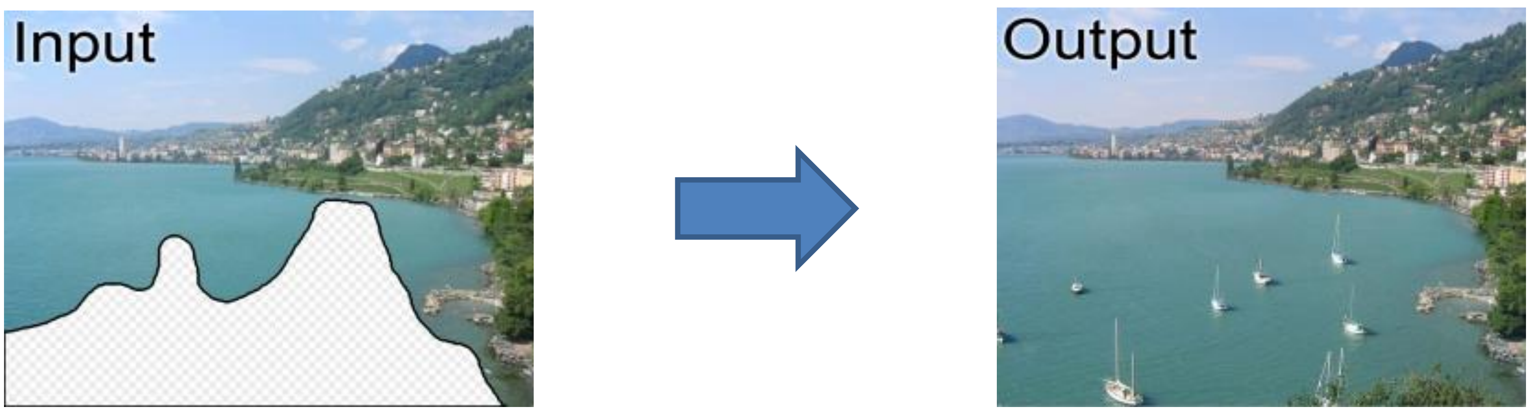


How can we leverage “Big Code”?

Scene Completion



Scene Completion



"Scene Completion Using Millions of Photographs", J. Hays, A. Efros, ACM SIGGRAPH 2007

Program Completion

```
Camera camera = Camera.open();
 camera.setDisplayOrientation(90);
```

Program Completion

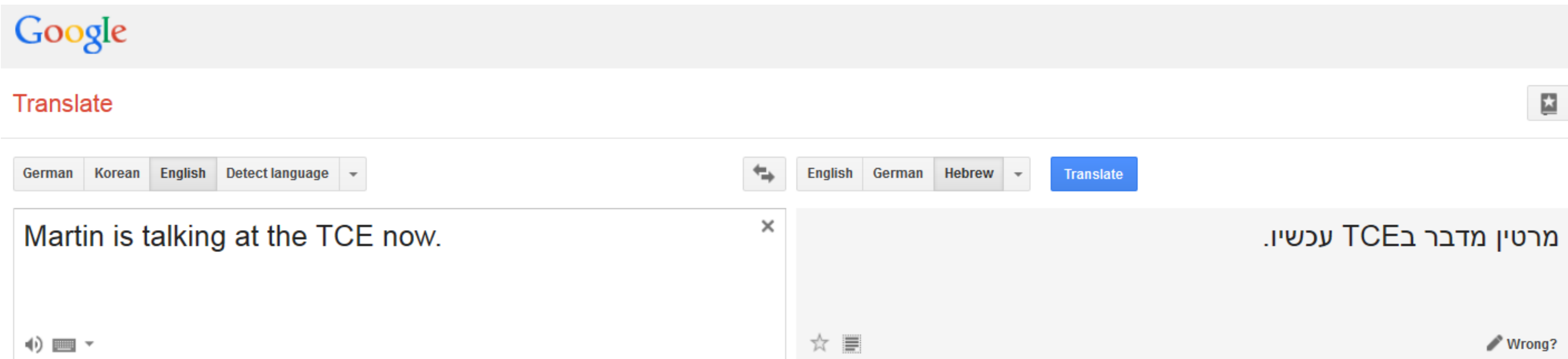
```
Camera camera = Camera.open();
 camera.setDisplayOrientation(90);
```

SLANG

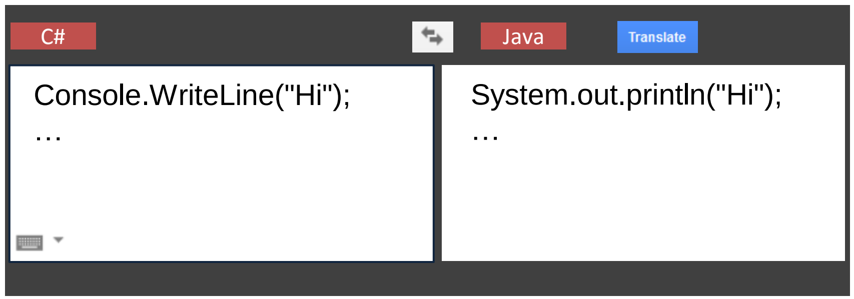
```
Camera camera = Camera.open();
 camera.setDisplayOrientation(90);

camera.unlock();

SurfaceHolder holder = getHolder();
holder.addCallback(this);
holder.setType(SurfaceHolder.STP);
MediaRecorder r = new MediaRecorder();

r.setCamera(camera);

r.setAudioSource(MediaRecorder.AS);
r.setVideoSource(MediaRecorder.VS);
r.setOutFormat(MediaRecorder.MPEG4);
```

"Code Completion with Statistical Language Models", V. Raychev, M.V., E. Yahav, **ACM PLDI'14**

Natural Language Translation



Programming Language Translation



“Phrase-based statistical translation of programming languages”, S. Karaivanov, V. Raychev, M. V., **Onward’14**

Image de-noisification

Program de-obfuscation

```
function FZ(e, t) {   var n = [];
var r = e.length;   var i = 0;
for (; i < r; i += t)      if (i + t <
r)      n.push(e.substring(i, i +
t)); else
n.push(e.substring(i, r));
return n;
}
```

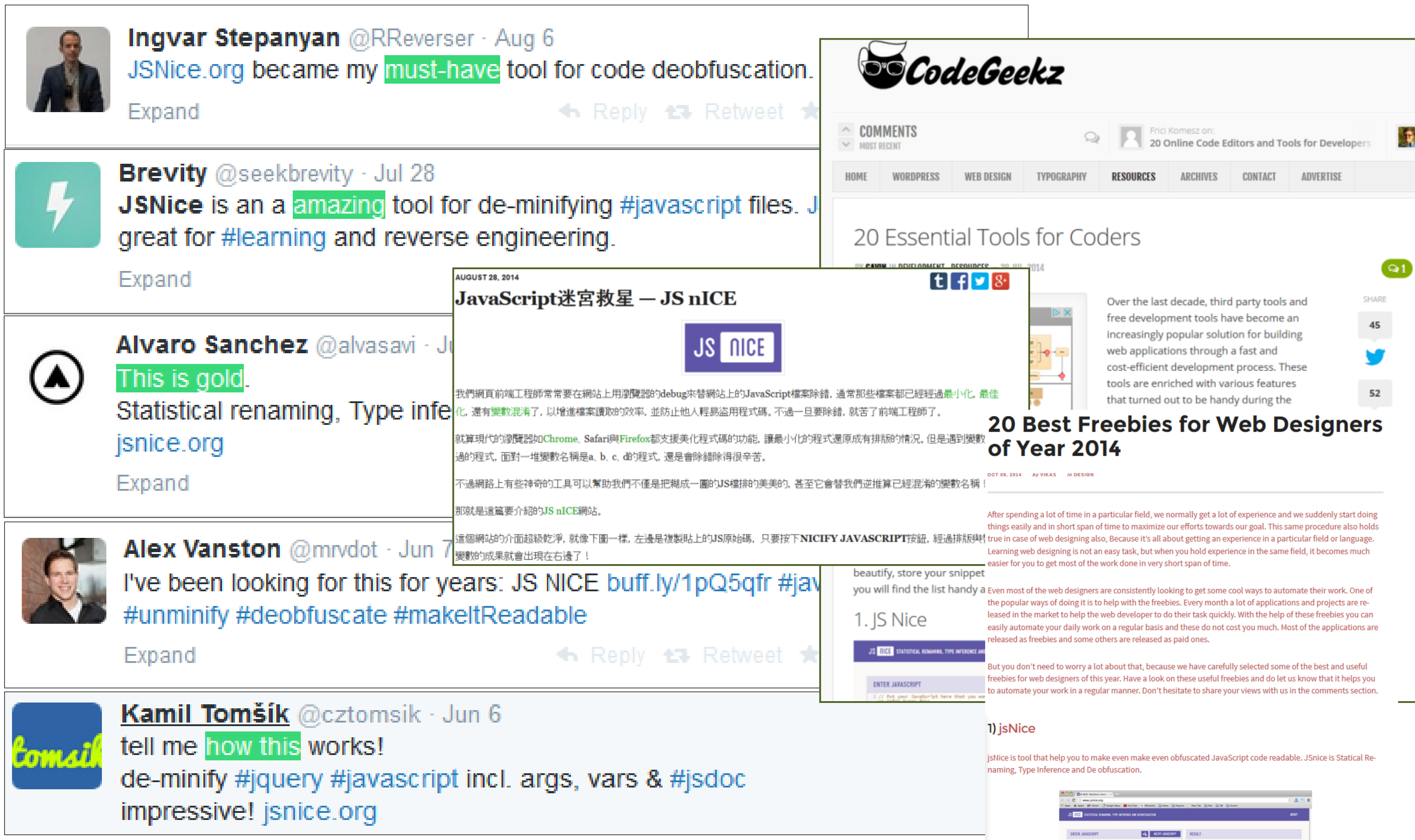
```
function chunkData(str, step) {
  var colNames = [];
  var len = str.length;
  var i = 0;
  for (; i < len; i += step)
    if (i + step < len)
      colNames.push(str.substring(i, i + step));
   else
      colNames.push(str.substring(i, len));
  return colNames;
}
```

Demo

"Predicting program properties from Big Code", V. Raychev, M. V., A. Krause, **ACM POPL'15**

JSNice.org: Impact

- ✓ Used in 191 countries
- ✓ 1,000+ Tweets
- ✓ Top ranked tool for JavaScript
- ✓ 30,000 users in 1st week



“Predicting program properties from Big Code”, V. Raychev, M. V., A. Krause, **ACM POPL’15**

Tutorial Outline

- ~~Motivation~~
  - ~~Potential applications~~

- Statistical language models
  - N-gram and Recurrent Networks, Smoothing
  - Application: code completion

- Graphical Models
  - Markov Networks, Conditional Random Fields
  - Inference in Markov Networks
  - Learning in Markov Networks
  - Application: predicting names and types

- Hands-on session with Nice2Predict

Probability Refresher

- (Discrete) probability distribution: $\sum_{x} P(x) = 1$
- For a joint probability distribution P(x,y), the distribution of a single variable, called the marginal, is: $\sum_{y} P(x,y) = 1$
- Conditional distribution: $P(y \mid x) = P(y, x) / P(x)$
- Bayes's rule: $P(y \mid x) = P(x \mid y)\ ) \times P(y) / P\ (x)$

Probability Refresher

- Independence: $P(x, y) = P(x)\,)\times\ P(y)$
  - written as: $x \perp y$

- Conditional Independence: $P(X, Y \mid Z) = P(X \mid Z) \times P(Y \mid Z)$
  - Set X is independent of set Y provided we know the value of variables in set Z.
  - written as $X \perp Y \mid Z$

Machine Learning for Programming

Applications	Code completion, Deobfuscation, Program synthesis, Feedback generation, Translation
Intermediate Representation	Sequences (sentences), Trees, Translation Table, Graphical Models (CRFs), Feature Vectors
Analyze Program (PL)	typestate analysis, scope analysis, control-flow analysis, alias analysis
Train Model (ML)	Neural Networks, N-gram language model, SVM, Structured SVM
Query Model (ML)	argmax P(y | x) y ∈ Ω; Greedy MAP inference

Motivation: Working with APIs

HIBERNATE

JFACE

Struts

spring source

eclipse PLUGIN CENTRAL

Apache Commons
http://commons.apache.org/

Apache ActiveMQ version 5.x

tapestry

SWT

GRAILS

Google play

jgraph. visualize everything

Statistical Code Completion: Capabilities

[V. Raychev, M. Vechev, E. Yahav ,PLDI'14]

```
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
```

?

```
MediaRecorder rec = new MediaRecorder();
```

?

```
rec.setAudioSource(MediaRecorder.AudioSource.MIC);
rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
```

?

```
rec.setOutputFile("file.mp4");
...
```

Statistical Code Completion: Capabilities

[V. Raychev, M. Vechev, E. Yahav ,PLDI'14]

```
Camera camera = Camera.open();
camera.setDisplayOrientation(90);

camera.unlock();

MediaRecorder rec = new MediaRecorder();

rec.setCamera(camera);

rec.setAudioSource(MediaRecorder.AudioSource.MIC);
rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);

rec.setAudioEncoder(1);
rec.setVideoEncoder(3);

rec.setOutputFile("file.mp4");
...
```

Statistical Code Completion: Capabilities

[V. Raychev, M. Vechev, E. Yahav ,PLDI'14]

```
Camera camera = Camera.open();
camera.setDisplayOrientation(90);

camera.unlock();

MediaRecorder rec = new MediaRecorder();

rec.setCamera(camera);

rec.setAudioSource(MediaRecorder.AudioSource.MIC);
rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);

rec.setAudioEncoder(1);
rec.setVideoEncoder(3);

rec.setOutputFile("file.mp4");
...
```

Handles multiple objects

Statistical Code Completion: Capabilities

[V. Raychev, M. Vechev, E. Yahav ,PLDI'14]

```
Camera camera = Camera.open();
camera.setDisplayOrientation(90);

camera.unlock();

MediaRecorder rec = new MediaRecorder();

rec.setCamera(camera);

rec.setAudioSource(MediaRecorder.AudioSource.MIC);
rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);

rec.setAudioEncoder(1);
rec.setVideoEncoder(3);

rec.setOutputFile("file.mp4");
...
```

Handles multiple objects

Infers multiple statements

Statistical Code Completion: Capabilities

[V. Raychev, M. Vechev, E. Yahav ,PLDI'14]

```
Camera camera = Camera.open();
camera.setDisplayOrientation(90);

camera.unlock();

MediaRecorder rec = new MediaRecorder();

rec.setCamera(camera);

rec.setAudioSource(MediaRecorder.AudioSource.MIC);
rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);

rec.setAudioEncoder(1);
rec.setVideoEncoder(3);

rec.setOutputFile("file.mp4");
...
```

Infers sequences not in training data

Handles multiple objects

Infers multiple statements

Key Insight

Regularities in code are similar to regularities in natural language

Key Insight

Regularities in code are similar to regularities in natural language

We want to learn that

```
MediaRecorder rec = new MediaRecorder();
```

is before

```
rec.setCamera(camera);
```

Key Insight

Regularities in code are similar to regularities in natural language

We want to learn that

```
MediaRecorder rec = new MediaRecorder();
```

is before

```
rec.setCamera(camera);
```

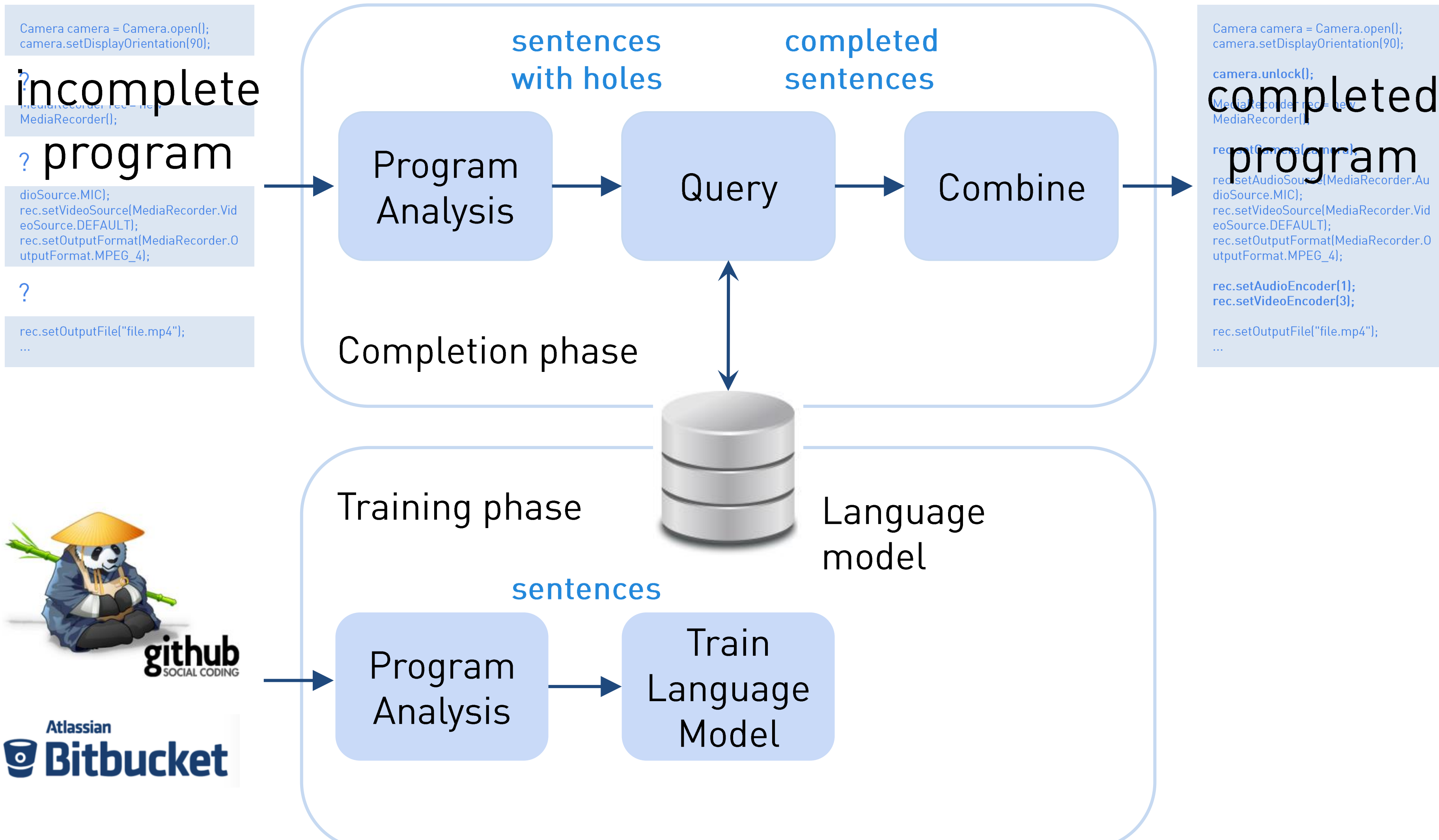
like in natural languages

Hello

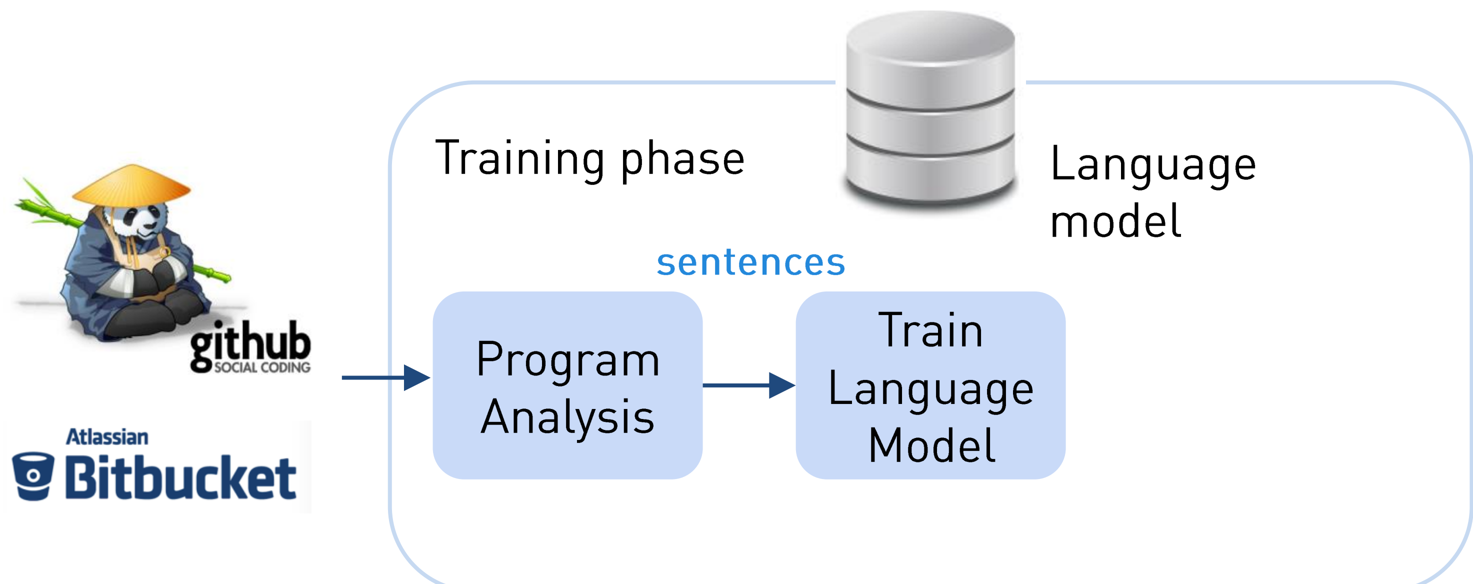
is before

World !

The SLANG System



The SLANG System



From Programs to Sentences

```
she = new X();
me = new Y();

me.sleep();
if (random()) {
   me.eat();
}
she.enter();
me.talk(she);
```

From Programs to Sentences

```
she = new X();
me = new Y();

me.sleep();
if (random()) {
   me.eat();
}
she.enter();
me.talk(she);
```

Typestate analysis

From Programs to Sentences

```
she = new X();
me = new Y();

me.sleep();
if (random()) {
  me.eat();
}
she.enter();
me.talk(she);
```

Typestate analysis
Alias analysis

From Programs to Sentences

```
she = new X();
me = new Y();

me.sleep();
if (random()) {
  me.eat();
}
she.enter();
me.talk(she);
```

Typestate analysis
Alias analysis

for abstract object **me**:

From Programs to Sentences

```
she = new X();
me = new Y();

me.sleep();
if (random()) {
  me.eat();
}
she.enter();
me.talk(she);
```

Typestate analysis

Alias analysis

for abstract object **me**:

Y_{init} sleep talk

From Programs to Sentences

```
she = new X();
me = new Y();

me.sleep();
if (random()) {
  me.eat();
}
she.enter();
me.talk(she);
```

Typestate analysis
Alias analysis

for abstract object **me**:

Y_{init} sleep talk
Y_{init} sleep eat talk

From Programs to Sentences

```
she = new X();
me = new Y();

me.sleep();
if (random()) {
  me.eat();
}
she.enter();
me.talk(she);
```

Typestate analysis
Alias analysis

for abstract object **me**:

Y_{init} sleep talk
Y_{init} sleep eat talk

for abstract object **she**:

From Programs to Sentences

```
she = new X();
me = new Y();

me.sleep();
if (random()) {
  me.eat();
}
she.enter();
me.talk(she);
```

Typestate analysis

Alias analysis

for abstract object **me**:

Y_{init} sleep talk

Y_{init} sleep eat talk

for abstract object **she**:

X_{init} enter $talk_{param1}$

Learn Regularities

Learn regularities in obtained sentences

Regularities in sentences ⇔ regularities in API usage

If we see many sequences like:

X_{init} enter $talk_{param1}$

then we should learn that $talk_{param1}$ is often after enter

Statistical Language Models

Given a sentence $s = w_1\ w_2\ w_3 \ldots\ w_n$

estimate $\mathbf{P}(\ w_1\ w_2\ w_3 \ldots\ w_n\ )$

Decomposed to conditional probabilities

$$\mathbf{P}(\ w_1\ w_2\ w_3 \ldots\ w_n\ ) = \prod_{i=1..n} \mathbf{P}(\ w_i \mid w_1\ \ldots\ w_{i-1}\ )$$

Statistical Language Models

Given a sentence $s = w_1\ w_2\ w_3 \ldots w_n$

estimate $\mathbf{P}(\ w_1\ w_2\ w_3 \ldots w_n\ )$

Decomposed to conditional probabilities

$$\mathbf{P}(\ w_1\ w_2\ w_3 \ldots w_n\ ) = \prod_{i=1..n} \mathbf{P}(\ w_i \mid w_1 \ldots w_{i-1}\ )$$

P(The quick brown fox jumped) =

P(The) **P**(quick | The) **P**(brown | The quick)

P(fox | The quick brown) **P**(jumped | The quick brown fox)

N-gram language model

Conditional probability only on previous n-1 words

$$\mathbf{P}(\ w_i \mid w_1 \ \dots \ w_{i-1}\ ) \approx \mathbf{P}(\ w_i \mid w_{i-n+1} \ \dots \ w_{i-1}\ )$$

N-gram language model

Conditional probability only on previous n-1 words

$$\mathbf{P}(w_i \mid w_1 \dots w_{i-1}) \approx \mathbf{P}(w_i \mid \underbrace{w_{i-n+1} \dots w_{i-1}}_{\text{n-1 words}})$$

N-gram language model

Conditional probability only on previous n-1 words

$$\mathbf{P}(\, w_i \mid w_1 \ldots w_{i-1}\,) \approx \mathbf{P}(\, w_i \mid \underbrace{w_{i-n+1} \ldots w_{i-1}}_{\text{n-1 words}}\,)$$

Training is achieved by counting n-grams.
E.g., with 3-gram language model, we get:

$$\mathbf{P}(\text{ jumped} \mid \text{The quick brown fox }) \approx \mathbf{P}(\text{ jumped} \mid \text{brown fox }) \approx \frac{\#(\text{ brown fox jumped })}{\#(\text{ brown fox })}$$

#(n-gram) - number of occurrences of n-gram in training data

Time complexity for each word encountered in training is constant, so training is usually fast.

Tri-gram language model

$$P(w_1 \cdot w_2 \cdot w_3 \cdot \ldots w_n) \sim= P(w_1) * P(w_2 \mid w_1) * P(w_3 \mid w_1 \cdot w_2) * \ldots * P(w_n \mid w_{n-2} \cdot w_{n-1})$$

3-grams	# of occurrences
brown fox jumped	125
brown fox walked	45
brown fox snapped	30

P (jumped | brown fox) ~= 125/200 ~= 0.625

P (brown fox jumped) ~=

$$\boxed{\begin{matrix}P(\text{brown}) \\ 200 / 600\end{matrix}} * \boxed{\begin{matrix}P(\text{fox} \mid \text{brown}) \\ 200 / 200\end{matrix}} * \boxed{\begin{matrix}P(\text{jumped} \mid \text{brown fox}) \\ 125 / 200\end{matrix}} \sim= 0.208$$

Key Problem: Sparsity of Data

What if this number is 0?

$$P(\text{ jumped } | \text{ brown fox }) \approx \frac{\#(\text{ brown fox jumped })}{\#(\text{ brown fox })}$$

The problem of sparsity gets worse as the size of the n-gram becomes larger.

We need to handle n-grams with 0 or few occurrences in the training data.
Techniques that can do that are: **smoothing**, **discounting**

Solution: Smoothing Techniques

- Smoothing techniques give non-zero probability to n-grams not in the training data.

- Essentially, they try to estimate how likely it is that the n-gram is missing due to the limited size of the training data.

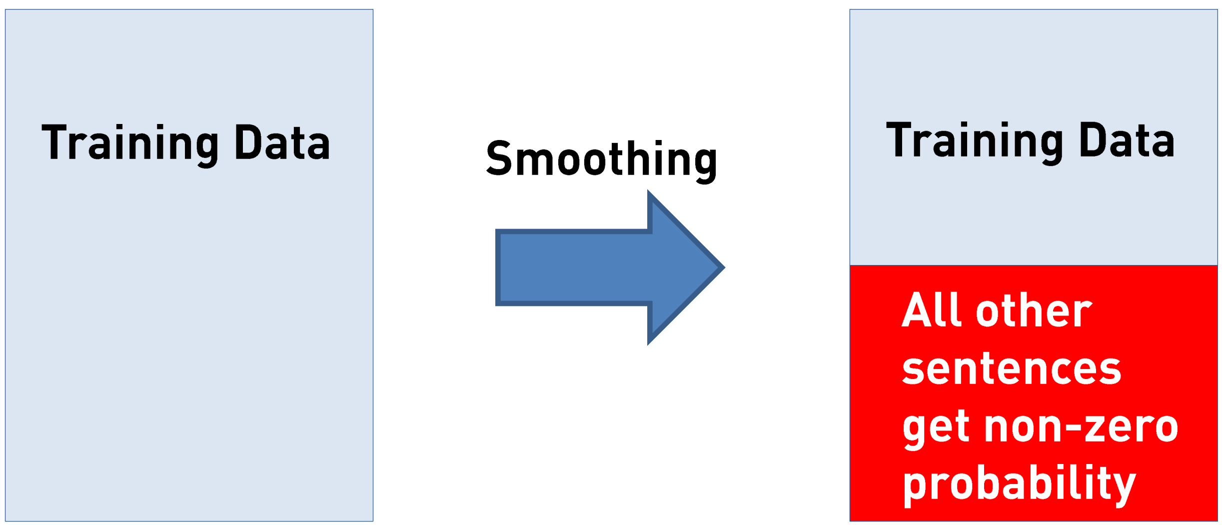
- They work by taking the probability mass of the existing n-grams and redistributing that mass over n-grams that occur zero times.

- Typically, probability of such n-grams is estimated by looking at the probabilities of n-1 grams , n-2 grams, unigrams, etc.

Smoothing: Intuitively

Distribution of probability mass before smoothing

Distribution of probability mass after smoothing



All other sentences get 0 probability

Querying the Probability Distribution

All scoring techniques that use smoothing follow this template:

$$P(w_i \mid w_{i-N+1}^{i-1}) = \begin{cases} P(w_i \mid w_{i-N+1}^{i-1}) & \text{If } w_{i-N+1}^{i} \text{ in training data} \\ B(w_{i-N+1}^{i-1}) \times P(w_i \mid w_{i-N+2}^{i-1}) & \text{otherwise} \end{cases}$$

Smoothing techniques differ by how they define this function

Example smoothing techniques are: Witten-Bell Interpolated (WBI), Witten-Bell Backoff (WBB), Natural Discounting (ND), Stupid-Backoff (SB)

Training: Witten-Bell Interpolated

$$P_{wbi}(w_i \mid w_{i-N+1}^{i-1}) = F_{wbi}(w_{i-N+1}^{i}) + B_{wbi}(w_{i-N+1}^{i-1}) \times P_{wbi}(w_i \mid w_{i-N+2}^{i-1})$$

$$F_{wbi}(w_{i-N+1}^{i}) = (1 - B_{wbi}(w_{i-N+1}^{i-1})) \times P_{ML}(w_i \mid w_{i-N+1}^{i-1})$$

$$B_{wbi}(w_{i-N+1}^{i-1}) = \frac{N(w_{i-N+1}^{i-1})}{N(w_{i-N+1}^{i-1}) + \#(w_{i-N+1}^{i-1})}$$

$$N(w_s) = |\{w : \#(w_s\ w) > 0\}|$$

Number of distinct followers of w_s
In the training data

LM Training Algorithm

1. Count all grams of length 1,..., N found in the training data

2. For each K-gram (0 < K < N) w_{i-K+1}^{i} in training data, calculate:

$$B_{\mathrm{wbi}}(w_{i-K+1}^{i-1}) \qquad P_{\mathrm{wbi}}(w_i \mid w_{i-K+1}^{i-1})$$

3. For the largest N-grams w_{i-N+1}^{i} in training data, calculate:

$$P_{\mathrm{wbi}}(w_i \mid w_{i-N+1}^{i-1})$$

N-gram language model

Conditional probability only on previous n-1 words

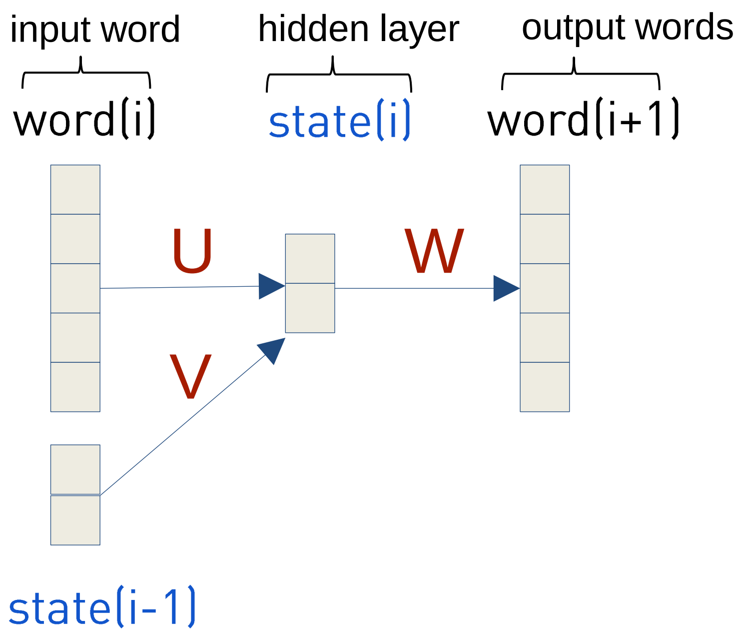
$$\mathbf{P}(\, w_i \mid w_1 \;\ldots\; w_{i-1}\,) \approx \mathbf{P}(\, w_i \mid \underbrace{w_{i-n+1} \;\ldots\; w_{i-1}}_{\text{n-1 words}}\,)$$

Existing library implementing language models is SRILM:

http://www.speech.sri.com/projects/srilm/

Recurrent Neural Networks (RNN)

RNNs can learn dependencies beyond the prior several words

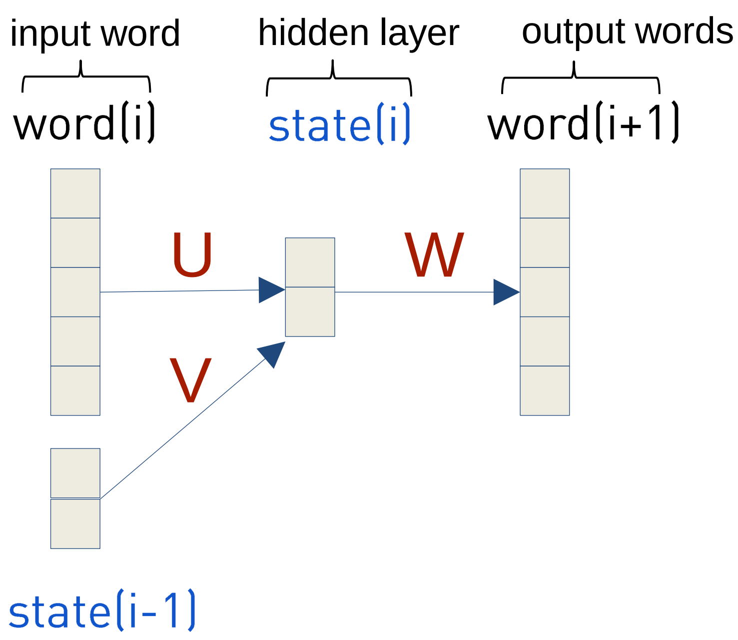


A neural network with internal state that stores probabilistic information about all previous words in a sentence With this, it can capture relationship between quick and jumped in:

P(jumped | The quick brown fox)

Recurrent Neural Networks (RNN)

RNNs can learn dependencies beyond the prior several words



A neural network with internal state that stores probabilistic information about all previous words in a sentence With this, it can capture relationship between quick and jumped in:

P(jumped | The quick brown fox)

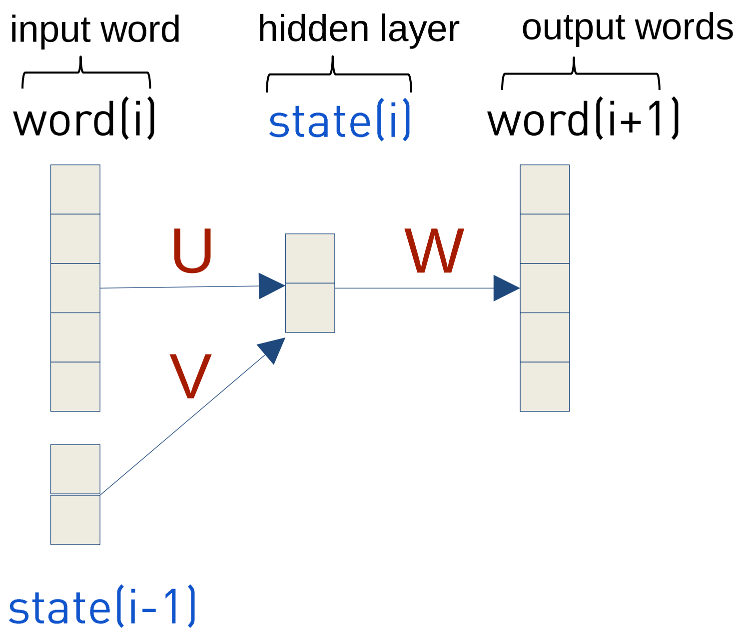
Matrices U, V and W learned through back propagation with time and some unfolding.

Time complexity for training RNNLM quadratic in size of vocabulary (for each word encountered in training), can be slow

Evaluation of a sentence s = $w_1 \cdot \ldots w_n$ proceeds on a word-by-word basis keeping the last s(t).

Recurrent Neural Networks (RNN)

RNNs can learn dependencies beyond the prior several words



RNNLM-40 Download:

http://rnnlm.org/

A neural network with internal state that stores probabilistic information about all previous words in a sentence With this, it can capture relationship between quick and jumped in:

P(jumped | The quick brown fox)

Matrices U, V and W learned through back propagation with time and some unfolding.

Time complexity for training RNNLM quadratic in size of vocabulary (for each word encountered in training), can be slow

Evaluation of a sentence s = $w_1 \cdot \ldots \cdot w_n$ proceeds on a word-by-word basis keeping the last s(t).

Code Completion

```
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
  list = smsMgr.divideMessage(message);
  ? {smsMgr, list} // (Hole H1)
} else {
  ? {smsMgr, message} // (Hole H2)
}
```

Code Completion

```
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
  list = smsMgr.divideMessage(message);
  ? {smsMgr, list} // (Hole H1)
} else {
  ? {smsMgr, message} // (Hole H2)
}
```

Abstract object **smsMgr**:

$\text{getDefault}_{\text{result}}$ divideMessage H1

$\text{getDefault}_{\text{result}}$ H2

Code Completion

```
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
  list = smsMgr.divideMessage(message);
  ? {smsMgr, list} // (Hole H1)
} else {
  ? {smsMgr, message} // (Hole H2)
}
```

Abstract object **smsMgr**:

$\text{getDefault}_{\text{result}}$ divideMessage H1

$\text{getDefault}_{\text{result}}$ H2

Code Completion

```
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
 list = smsMgr.divideMessage(message);
  ? {smsMgr, list} // (Hole H1)
} else {
  ? {smsMgr, message} // (Hole H2)
}
```

Abstract object **smsMgr**:

$\text{getDefault}_{\text{result}}$ divideMessage H1

$\text{getDefault}_{\text{result}}$ H2

Abstract object **list**:

$\text{divideMessage}_{\text{result}}$ H1

Code Completion

```
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
  list = smsMgr.divideMessage(message);
  ? {smsMgr, list} // (Hole H1)
} else {
  ? {smsMgr, message} // (Hole H2)
}
```

Abstract object **smsMgr**:

$\text{getDefault}_{\text{result}}$ divideMessage H1

$\text{getDefault}_{\text{result}}$ H2

Abstract object **list**:

$\text{divideMessage}_{\text{result}}$ H1

Code Completion

```
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
  list = smsMgr.divideMessage(message);
  ? {smsMgr, list} // (Hole H1)
} else {
  ? {smsMgr, message} // (Hole H2)
}
```

Abstract object **smsMgr**:

$\text{getDefault}_{\text{result}}$ divideMessage H1

$\text{getDefault}_{\text{result}}$ H2

Abstract object **list**:

$\text{divideMessage}_{\text{result}}$ H1

Abstract object **message**:

length $\text{divideMessage}_{\text{param1}}$

length H2

Code Completion

```
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
  list = smsMgr.divideMessage(message);
  ? {smsMgr, list} // (Hole H1)
} else {
  ? {smsMgr, message} // (Hole H2)
}
```

Abstract object **smsMgr**:

getDefault$_{result}$ divideMessage H1

getDefault$_{result}$ H2

Abstract object **list**:

divideMessage$_{result}$ H1

Abstract object **message**:

length divideMessage$_{param1}$

length H2

Code Completion

$\text{getDefault}_{\text{result}}$ divideMessage **H1**	
$\text{getDefault}_{\text{result}}$ **H2**	
$\text{divideMessage}_{\text{result}}$ **H1**	
length **H2**	

Code Completion

	Completion probability
$\text{getDefault}_{\text{result}}$ divideMessage **sendMultipartTextMessage**	0.0033
$\text{getDefault}_{\text{result}}$ divideMessage **sendTextMessage**	0.0016

$\text{getDefault}_{\text{result}}$ **sendTextMessage**	0.0073
$\text{getDefault}_{\text{result}}$ **sendMultipartTextMessage**	0.0010

$\text{divideMessage}_{\text{result}}$ $\textbf{sendMultipartTextMessage}_{\textbf{param3}}$	0.0821

length **length**	0.0132
length **split**	0.0080
length $\textbf{sendTextMessage}_{\textbf{param3}}$	0.0017

Code Completion

	Completion probability
$\text{getDefault}_{\text{result}}$ divideMessage **sendMultipartTextMessage**	0.0033
$\text{getDefault}_{\text{result}}$ divideMessage **sendTextMessage**	0.0016

$\text{getDefault}_{\text{result}}$ **sendTextMessage**	0.0073
$\text{getDefault}_{\text{result}}$ **sendMultipartTextMessage**	0.0010

$\text{divideMessage}_{\text{result}}$ $\textbf{sendMultipartTextMessage}_{\textbf{param3}}$	0.0821

length **length**	0.0132
length **split**	0.0080
length $\textbf{sendTextMessage}_{\textbf{param3}}$	0.0017

Code Completion

	Completion probability
getDefault$_{result}$ divideMessage **sendMultipartTextMessage**	0.0033
getDefault$_{result}$ divideMessage **sendTextMessage**	0.0016
getDefault$_{result}$ **sendTextMessage**	0.0073
getDefault$_{result}$ **sendMultipartTextMes**	
divideMessage$_{result}$ **sendMultipartTextMes**	
length **length**	
length **split**	0.0080
length **sendTextMessage**$_{param3}$	0.0017

Not a feasible solution: completions disagree on selected method

The solution must satisfy program constraints

Code Completion

	Completion probability
$\text{getDefault}_{\text{result}}$ divideMessage **sendMultipartTextMessage**	0.0033
$\text{getDefault}_{\text{result}}$ divideMessage **sendTextMessage**	0.0016
$\text{getDefault}_{\text{result}}$ **sendTextMessage**	0.0073
$\text{getDefault}_{\text{result}}$ **sendMultipartTextMessage**	0.0010
$\text{divideMessage}_{\text{result}}$ **sendMultipartTextMessage**$_{\textbf{param3}}$	0.0821
length **length**	0.0132
length **split**	0.0080
length **sendTextMessage**$_{\textbf{param3}}$	0.0017

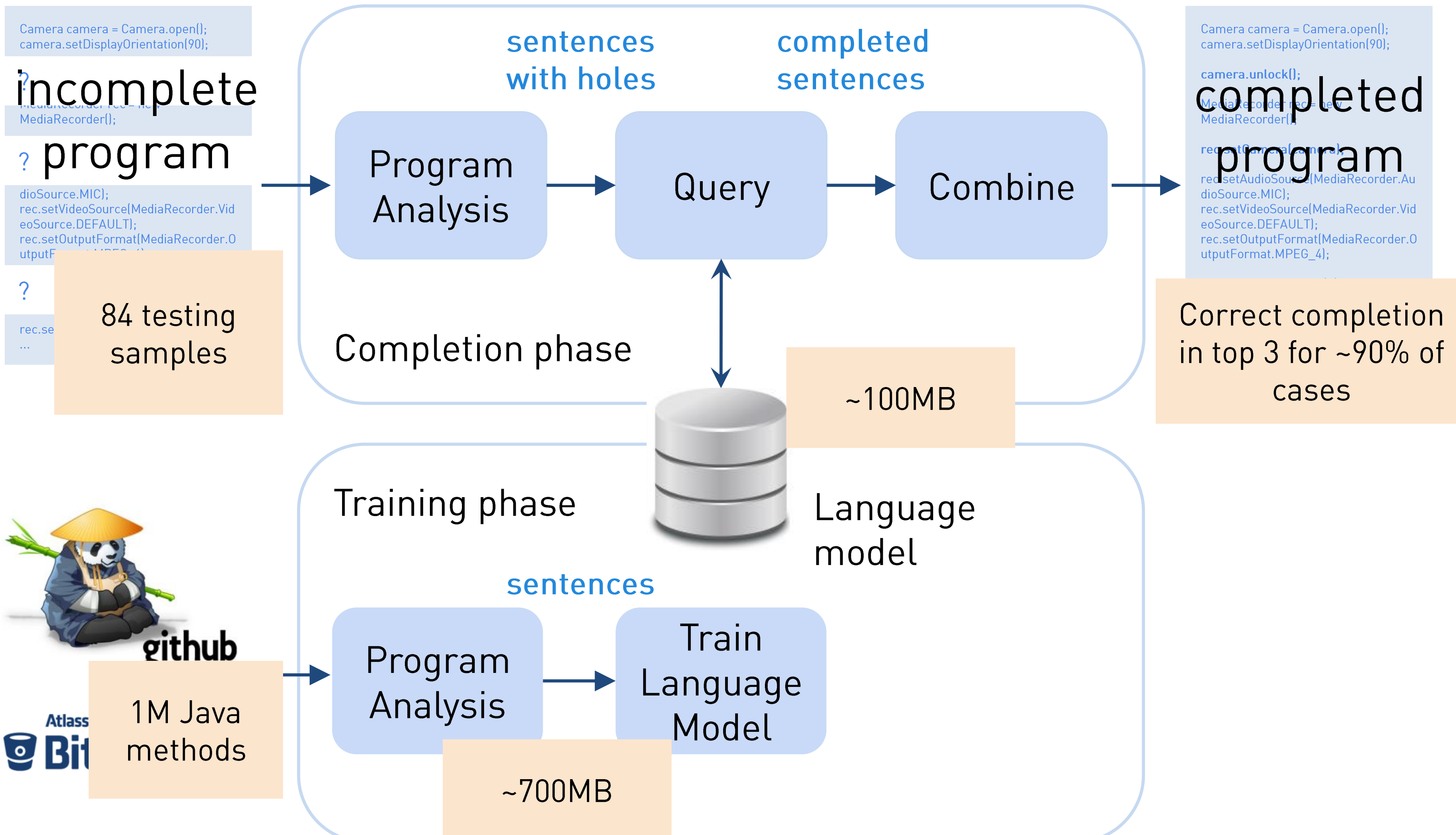
Code Completion

	Completion probability
getDefault$_{result}$ divideMessage **sendMultipartTextMessage**	0.0033
getDefault$_{result}$ divideMessage **sendTextMessage**	0.0016
getDefault$_{result}$ **sendTextMessage**	0.0073
getDefault$_{result}$ **sendMultipartTextMessage**	0.0010
divideMessage$_{result}$ **sendMultipartTextMessage$_{param3}$**	0.0821
length **length**	0.0132
length **split**	0.0080
length **sendTextMessage$_{param3}$**	0.0017

Code Completion

```
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
  list = smsMgr.divideMessage(message);
  smsMgr.sendMultipartTextMessage(...list...);
} else {
  smsMgr.sendTextMessage(...message...);
}
```

The SLANG System



Lessons

- Language Models
  - Easy and cheap to train
  - Important to deal with sparsity via smoothing
  - Connection to PL: decide what to ``compile'' into the word
  - Program Analysis need not be sound
  - Can be tricky to capture long distance relationships

- Recurrent Networks
  - Can capture longer distance relationships
  - More expensive to train
  - Experimentally similar to language models

Tutorial Outline

- ~~Motivation~~
  - ~~Potential applications~~

- ~~Statistical language models~~
  - ~~N-gram and Recurrent Networks, Smoothing~~
  - ~~Application: code completion~~

- Graphical Models
  - Markov Networks, Conditional Random Fields
  - Inference in Markov Networks
  - Learning in Markov Networks
  - Application: predicting names and types

- Hands-on session with Nice2Predict

Machine Learning for Programming

Applications	Code completion, Deobfuscation, Program synthesis, Feedback generation, Translation
Intermediate Representation	Sequences (sentences), Trees, Translation Table, Graphical Models (CRFs), Feature Vectors
Analyze Program (PL)	typestate analysis, scope analysis, control-flow analysis, alias analysis
Train Model (ML)	Neural Networks, N-gram language model, SVM, Structured SVM
Query Model (ML)	$\underset{y \in \Omega}{\text{argmax}}\ P(y \mid x)$, Greedy MAP inference

More information and tutorials at:

http://www.nice2predict.org/
http://www.srl.inf.ethz.ch/

Machine Learning for Programming

Applications	Code completion, Deobfuscation, Program synthesis, Feedback generation, Translation
Intermediate Representation	Sequences (sentences), Trees, Translation Table, Graphical Models (CRFs), Feature Vectors
Analyze Program (PL)	typestate analysis, scope analysis, control-flow analysis, alias analysis
Train Model (ML)	Neural Networks, N-gram language model, SVM, Structured SVM
Query Model (ML)	argmax P(y | x), y ∈ Ω; Greedy MAP inference

More information and tutorials at:
http://www.nice2predict.org/
http://www.srl.inf.ethz.ch/

Machine Learning for Programming

Applications	Code completion, Deobfuscation, Program synthesis, Feedback generation, Translation	
Intermediate Representation	Sequences (sentences), Translation Table, Trees	Graphical Models (CRFs), Feature Vectors
Analyze Program (PL)	typestate analysis, scope analysis, control-flow analysis	alias analysis
Train Model (ML)	Neural Networks, SVM, N-gram language model	Structured SVM
Query Model (ML)	argmax $P(y \mid x)$ $y \in \Omega$	Greedy MAP inference

NICE 2 Predict

More information and tutorials at:

http://www.nice2predict.org/
http://www.srl.inf.ethz.ch/

Goal

```
function f(a) {
  var b = document.getElementById(a);
  return b;
}
```

Goal

```
function f(a) {
  var b = document.getElementById(a);
  return b;
}
```

unknown facts: a b

known facts: f document getElementById

Goal

```
function f(a) {
  var b = document.getElementById(a);
  return b;
}
```

unknown facts: a b

known facts: f document getElementById

Predict unknown facts given some known facts

Challenges

Facts to be predicted are dependent

Millions of candidate choices

Must quickly learn from huge codebases

Prediction should be fast (real time)

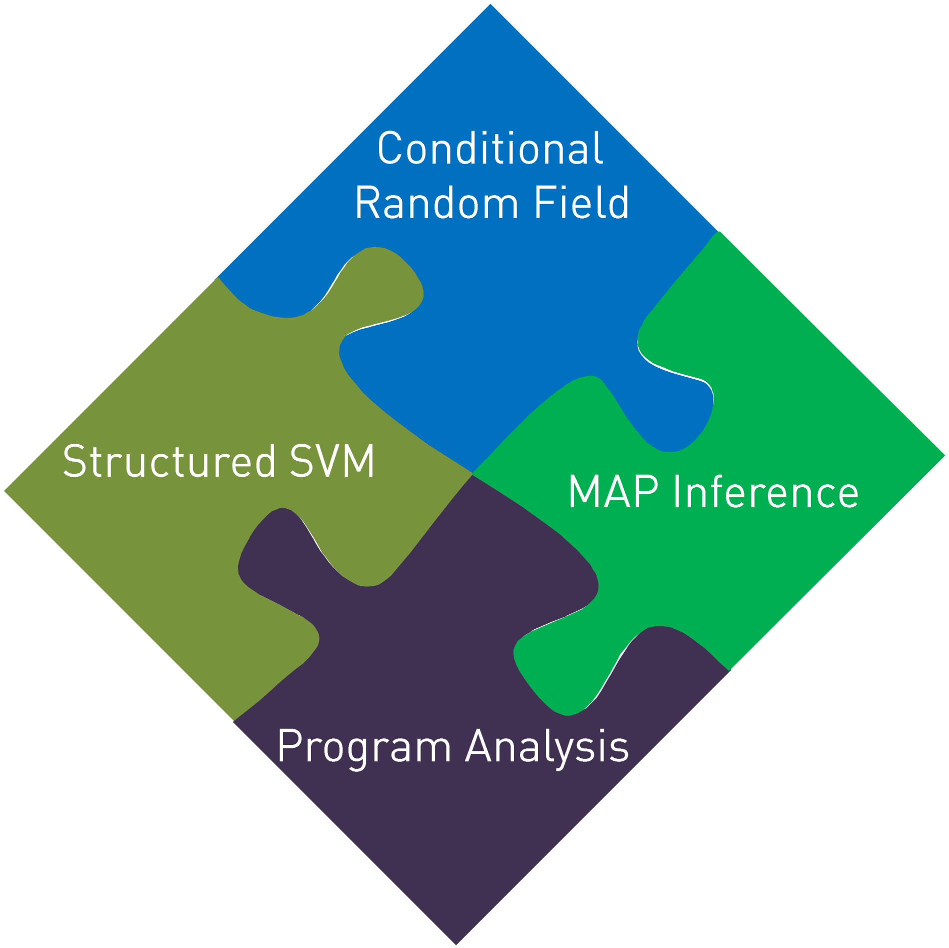
Key Idea

Phrase the problem of predicting program facts as

Structured Prediction for Programs

Structured Prediction for Programs

[V. Raychev, M. V., A. Krause, ACM POPL'15]



First connection between Programs and Conditional Random Fields

Factors

- A factor (or potential) φ is a function from a set of random variables **D** to a real number **R**

$$\varphi\colon \mathbf{D} \rightarrow \mathbf{R}$$

- The set of variables **D** is the scope of the factor φ
  - we are typically concerned with non-negative factors

- Intuition: captures affinity, agreement, compatibility of the variables in **D**

Factors: Example

- Assume we have 4 (binary) random variables:
  - Alice, Bob, Ceco and Dobri
- Example factors
  - φ_1 (0,0) = 30 says we believe Alice and Bob agree on 0 with belief 30
  - φ_3 (C,D) says that Ceco and Dobri argue all the time ☺

φ_1 (A,B)

A	B	value
0	0	30
0	1	5
1	0	1
1	1	10

φ_2 (B,C)

B	C	value
0	0	100
0	1	1
1	0	1
1	1	100

φ_3 (C,D)

C	D	value
0	0	1
0	1	100
1	0	100
1	1	1

φ_4 (D,A)

D	A	value
0	0	100
0	1	1
1	0	1
1	1	100

Factors: Example

- Assume we have 4 (binary) random variables:
  - Alice, Bob, Ceco and Dobri
- Example factors
  - φ_1 (0,0) = 30 says we believe Alice and Bob agree on 0 with belief 30
  - φ_3 (C,D) says that Ceco and Dobri argue all the time ☺

φ_1 (A,B)

A	B	value
0	0	30
0	1	5
1	0	1
1	1	10

φ_2 (B,C)

B	C	value
0	0	100
0	1	1
1	0	1
1	1	100

φ_3 (C,D)

C	D	value
0	0	1
0	1	100
1	0	100
1	1	1

φ_4 (D,A)

D	A	value
0	0	100
0	1	1
1	0	1
1	1	100

Key point: (Local) factors have no probabilistic interpretation

Defining a Global Probabilistic Model

Joint probability distribution

$$P(A, B, C, D) \quad = \quad F(A,B,C,D) \;/\; Z(A,B,C,D)$$

$$F(A, B, C, D) \quad = \quad \varphi_1(A,B) \times \varphi_2(B,C) \times \varphi_3(C,D) \times \varphi_4(D,A)$$

$$Z(A,B,C,D) \quad = \quad \sum_{\substack{a \in A,\, b \in B \\ c \in C,\, d \in D}} F(a, b, c, d)$$

Partition function

P(A,B,C,D)

A	B	C	D	Unnormalized (F)	Normalized (P = F/Z)
0	0	0	0	300,000	0.04
0	0	0	1	300,000	0.04
0	0	1	0	300,000	0.04
0	0	1	1	30	4.1×10^{-6}
…	…	…	…	…	…

Z(A,B,C,D) = 7,201,840

P(A,B,C,D)

A	B	C	D	Unnormalized (F)	Normalized (P = F/Z)
0	0	0	0	300,000	0.04
0	0	0	1	300,000	0.04
0	0	1	0	300,000	0.04
0	0	1	1	30	4.1×10^{-6}
…	…	…	…	…	…

$Z(A,B,C,D) = 7{,}201{,}840$

We can now answer probability queries on P(A, B, C, D):

For example: $P(B = 1) = \sum_{\substack{a \in A \\ c \in C \\ d \in D}} P(A, 1, C, D) \sim= 0.732$

Markov Network

Let $X = \{X_1, X_2, \ldots, X_n\}$ be a set of random variables and $D_1, D_2, \ldots, D_m \subseteq X$

Then, a Markov Network is defined as:

$$P(X_1, X_2, \ldots, X_n) = \frac{\varphi_1 (D_1) \times \varphi_2 (D_2) \times \ldots \times \varphi_m (D_m)}{Z(X_1, X_2, \ldots, X_n)}$$

$$Z(X_1, X_2, \ldots, X_n) = \sum \varphi_1 (D_1) \times \varphi_2 (D_2) \times \ldots \times \varphi_m (D_m)$$

Markov Network

Let $X = \{X_1, X_2, \ldots, X_n\}$ be a set of random variables and $D_1, D_2, \ldots, D_m \subseteq X$

Then, a Markov Network is defined as:

$$P(X_1, X_2, \ldots, X_n) = \frac{\varphi_1(D_1) \times \varphi_2(D_2) \times \ldots \times \varphi_m(D_m)}{Z(X_1, X_2, \ldots, X_n)}$$

$$Z(X_1, X_2, \ldots, X_n) = \sum \varphi_1(D_1) \times \varphi_2(D_2) \times \ldots \times \varphi_m(D_m)$$

If the factors are strictly positive, then P is called a Gibbs distribution

If the domains of all factors is of size 2, the network is called pairwise

Graphs vs. Probability Distributions

We will next relate graphs and probability distributions. This is important for two reasons.

First, it allows us to determine properties (e.g., independence of variables) of a probability distribution directly from the graph.

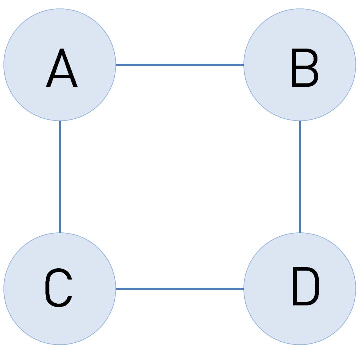
Second, it allows us to answer queries (e.g., MAP inference) on the probability distribution by working with the graph.

Graph Factorization

A distribution P equipped with a set of factors φ_1 (D_1) , … , φ_m (D_m) factorizes over a graph G if each D_i is a complete subgraph of G

Graph Factorization

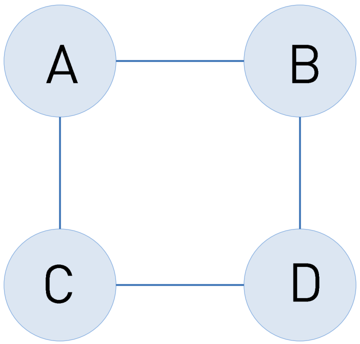
A distribution P equipped with a set of factors φ_1 (D_1) , … , φ_m (D_m) factorizes over a graph G if each D_i is a complete subgraph of G



$$P(A, B, C, D) =$$
$$\varphi_1 (A, B) \times \varphi_2 (B, D) \times \varphi_3 (D, C) \times \varphi_4 (C, A) / Z$$

Graph Factorization

A distribution P equipped with a set of factors φ_1 (D_1) , … , φ_m (D_m) factorizes over a graph G if each D_i is a complete subgraph of G

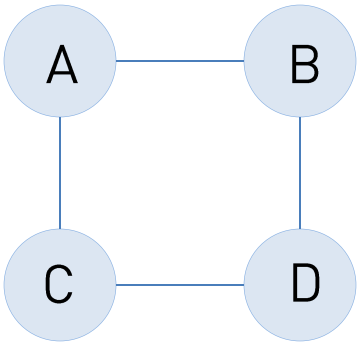


$$P(A, B, C, D) =$$
$$\varphi_1 (A, B) \times \varphi_2 (B, D) \times \varphi_3 (D, C) \times \varphi_4 (C, A) / Z$$

Here, P factorizes over G

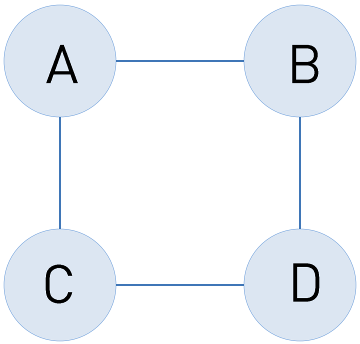
Graph Factorization

A distribution P equipped with a set of factors φ_1 (D_1) , … , φ_m (D_m) factorizes over a graph G if each D_i is a complete subgraph of G



$$P(A, B, C, D) = \varphi_1 (A, B) \times \varphi_2 (B, D) \times \varphi_3 (D, C) \times \varphi_4 (C, A) / Z$$

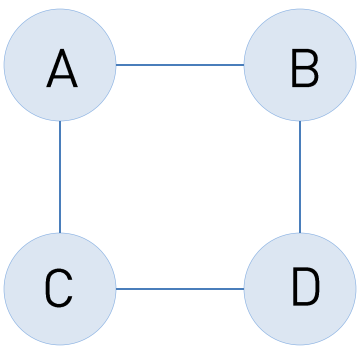
Here, P factorizes over G

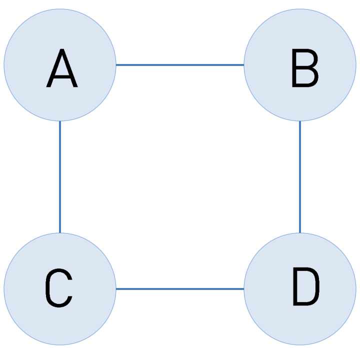


$$P(A, B, C, D) = \varphi_1 (A, B, C) \times \varphi_2 (B, D, C) / Z$$

Graph Factorization

A distribution P equipped with a set of factors φ_1 (D_1) , ... , φ_m (D_m) factorizes over a graph G if each D_i is a complete subgraph of G





P(A, B, C, D) =
φ_1 (A, B) $\times$ φ_2 (B, D) $\times$ φ_3 (D, C) $\times$ φ_4 (C, A) / Z

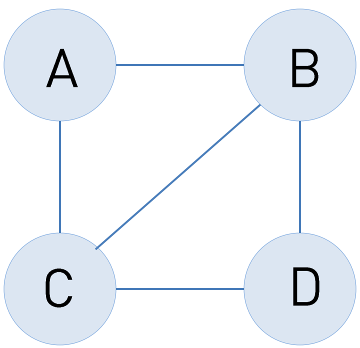
Here, P factorizes over G

P(A, B, C, D) =
φ_1 (A, B, C) $\times$ φ_2 (B, D, C) / Z

Here, P does not factorize over G

Graph Factorization

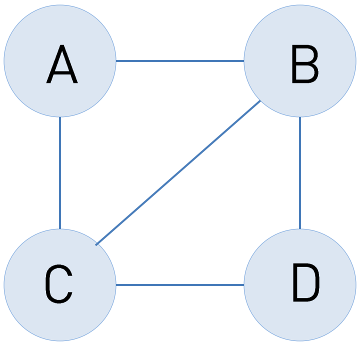
A distribution P equipped with a set of factors φ_1 (D_1) , … , φ_m (D_m)
factorizes over a graph G if each D_i is a complete subgraph of G



$P(A, B, C, D) = \varphi_1 (A, B, C) \times \varphi_2 (B, C, D) / Z$

Graph Factorization

A distribution P equipped with a set of factors φ_1 (D_1) , … , φ_m (D_m) factorizes over a graph G if each D_i is a complete subgraph of G

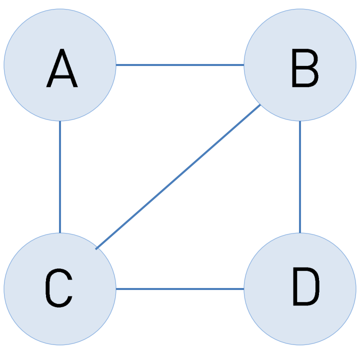


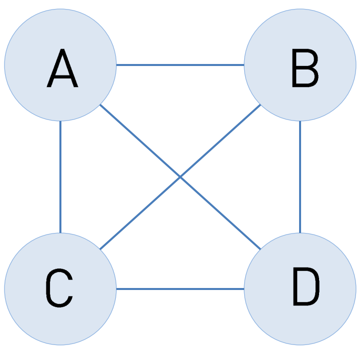
$$P(A, B, C, D) = \varphi_1 (A, B, C) \times \varphi_2 (B, C, D) / Z$$

Here, P factorizes over G

Graph Factorization

A distribution P equipped with a set of factors φ_1 (D_1) , … , φ_m (D_m) factorizes over a graph G if each D_i is a complete subgraph of G





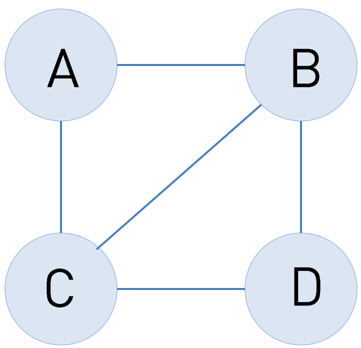
$P(A, B, C, D) = \varphi_1 (A, B, C) \times \varphi_2 (B, C, D) / Z$

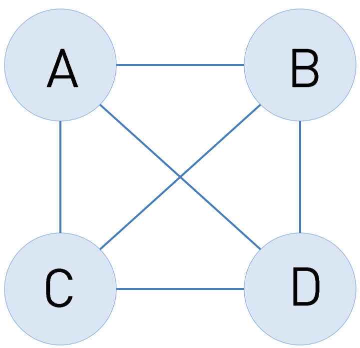
$P(A, B, C, D) = \varphi_1 (A, B, C, D) / Z$

Here, P factorizes over G

Graph Factorization

A distribution P equipped with a set of factors $\varphi_1(D_1), \ldots, \varphi_m(D_m)$ factorizes over a graph G if each D_i is a complete subgraph of G





P(A, B, C, D) $= \varphi_1(A, B, C) \times \varphi_2(B, C, D) / Z$

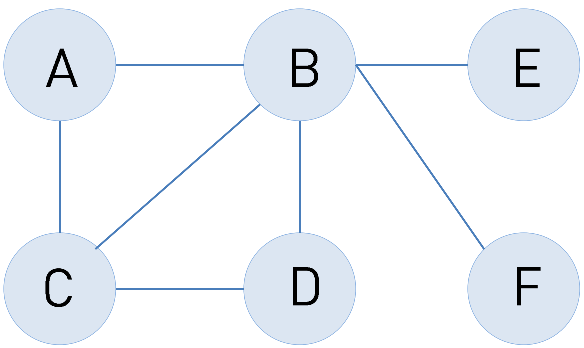
Here, P factorizes over G

P(A, B, C, D) $= \varphi_1(A, B, C, D) / Z$

Here, P factorizes over G

Graph Factorization

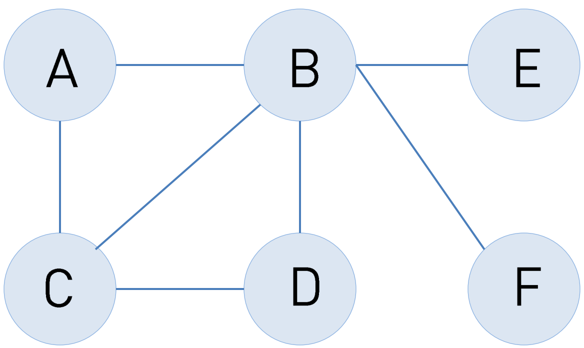
A distribution P equipped with a set of factors $\varphi_1 (D_1), \ldots, \varphi_m (D_m)$ factorizes over a graph G if each D_i is a complete subgraph of G



$$P(A, B, C, D) = \varphi_1 (A, B, C) \times \varphi_2 (B, C, D) \times \varphi_3 (B, E) \times \varphi_4 (B, F) / Z$$

Graph Factorization

A distribution P equipped with a set of factors φ_1 (D_1) , … , φ_m (D_m) factorizes over a graph G if each D_i is a complete subgraph of G



$$P(A, B, C, D) = \varphi_1 (A, B, C) \times \varphi_2 (B, C, D) \times \varphi_3 (B, E) \times \varphi_4 (B, F) / Z$$

Here, P factorizes over G

Graphs vs. Probabilities

There is an important connection between a probability distribution that factorizes over a graph and the properties of the graph.

In particular, we can discover various independence properties of the probabilistic distribution directly from the graph

Graph Separation

Two sets of disjoint nodes A and B in a graph G are separated by a set S, if every path between A and B goes through a node in S. If there is no path between A and B then A and B are separated. A and B are also separated if $S = \emptyset$ and no path between A and B exists.

The Global Markov Property says that if A and B are separated by S, then

$$A \perp B \mid S \quad \text{that is} \quad P(A, B \mid S) = P(A \mid S) \times P(B \mid S)$$

Graph Separation

Two sets of disjoint nodes A and B in a graph G are separated by a set S, if every path between A and B goes through a node in S. If there is no path between A and B then A and B are separated. A and B are also separated if $S = \varnothing$ and no path between A and B exists.

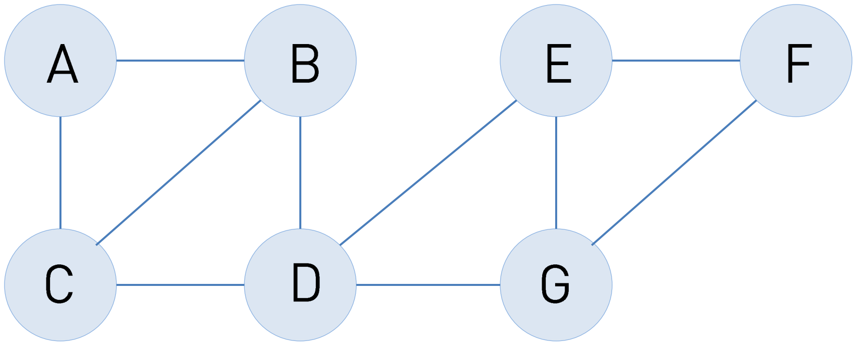
The Global Markov Property says that if A and B are separated by S, then

$$A \perp B \mid S \quad \text{that is} \quad P(A, B \mid S) = P(A \mid S) \times P(B \mid S)$$

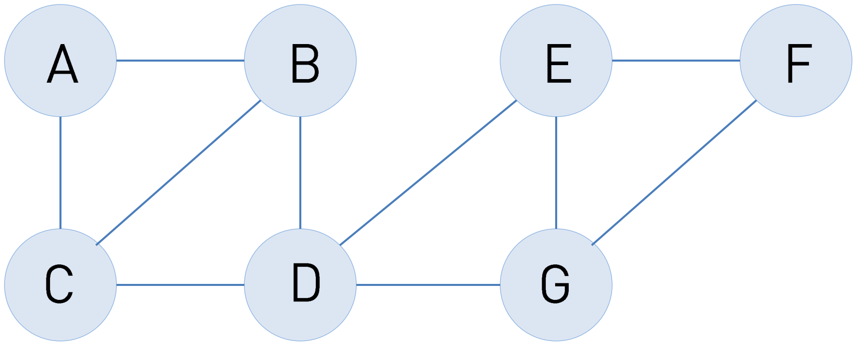
An algorithm to determine all global independencies $A \perp B \mid S$ is:

1. Given S, remove all edges adjacent to nodes in S resulting in a new graph G'
2. Check if a path between nodes in A and B exists in G'

Graph Separation: Example



Graph Separation: Example



Example of global independencies found in the above graph:

$$\{A\} \perp \{G\} \mid \{D\}$$

$$\{B\} \perp \{F\} \mid \{E,G\}$$

Independence Assertions

We use the notation **I(P)** to denote the set of conditional independence assertions of the form $X \perp Y \mid Z$ for the probability distribution P.

Similarly, we use **I(G)** to denote the set of assertions of the above kind exhibited by a graph G – these are called the Global Markov assertions

G is called an I-map of P if $\mathbf{I(G)} \subseteq \mathbf{I(P)}$

Independence Assertions

We use the notation **I(P)** to denote the set of conditional independence assertions of the form $X \perp Y \mid Z$ for the probability distribution P.

Similarly, we use **I(G)** to denote the set of assertions of the above kind exhibited by a graph G – these are called the Global Markov assertions

G is called an I-map of P if **I(G)** $\subseteq$ **I(P)**

For positive distributions (every value in the range is > 0), we have:

Theorem: P factorizes over G iff **I(G)** $\subseteq$ **I(P)**

If the distribution is not positive, then there is a counter example to the above theorem in one direction.

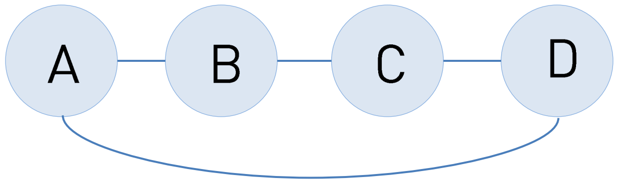
Counter-example to IFF

Consider the distribution **P** over 4 variables A, B, C and D where:

P(0, 0, 0, 0) = 1/8	P(1, 0, 0, 0) = 1/8	P(1, 1, 0, 0) = 1/8	P(1, 1, 1, 0) = 1/8
P(0, 0, 0, 1) = 1/8	P(0, 0, 1, 1) = 1/8	P(0, 1, 1, 1) = 1/8	P(1, 1, 1, 1) = 1/8

Here, P is not a positive distribution as for all other values of A,B,C,D, it is 0

Consider the graph **G**:



I(G):

$\{A\} \perp \{C\} \mid \{B, D\}$

$\{B\} \perp \{D\} \mid \{A, C\}$

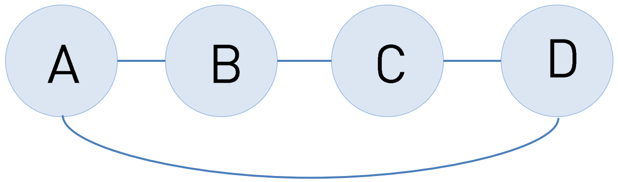
Counter-example to IFF

Consider the distribution **P** over 4 variables A, B, C and D where:

P(0, 0, 0, 0) = 1/8	P(1, 0, 0, 0) = 1/8	P(1, 1, 0, 0) = 1/8	P(1, 1, 1, 0) = 1/8
P(0, 0, 0, 1) = 1/8	P(0, 0, 1, 1) = 1/8	P(0, 1, 1, 1) = 1/8	P(1, 1, 1, 1) = 1/8

Here, P is not a positive distribution as for all other values of A,B,C,D, it is 0

Consider the graph **G**:



I(G):

$\{A\} \perp \{C\} \mid \{B, D\}$

$\{B\} \perp \{D\} \mid \{A, C\}$

Here we have that I(G) $\subseteq$ I(P), yet **P** does not factorize according to G

Independence Assertions on Graphs

Pairwise Independence:

$$I_p(G) = \{(X_1 \perp X_2 \mid X \setminus \{X_1, X_2\}) : \text{edge}(X_1, X_2) \notin G\}$$

Local Independence:

$$I_l(G) = \{X_1 \perp X - \{X_1\} - MB_G(X_1) \mid MB_G(X_1)\}$$

$MB_G(X_1)$ is the Markov Blanket of node X_1 : the neighbors of node X_1 in G

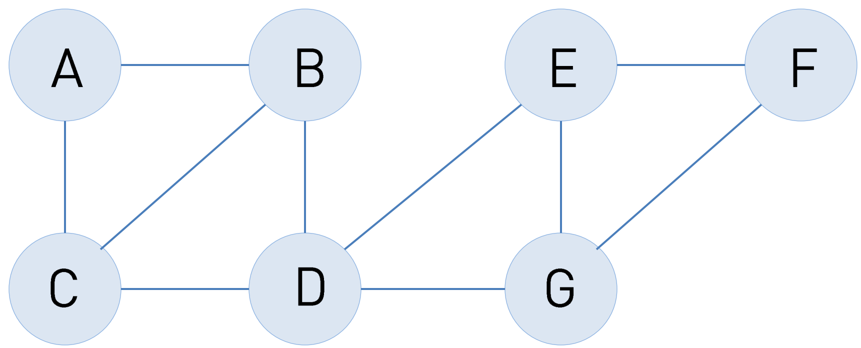
For positive distributions, the following statements are equivalent:

$$P \models I_p(G) \qquad P \models I_l(G) \qquad P \models I(G)$$

For **general** distributions, we have a weaker relationship:

$$\text{If } P \models I(G) \text{ then } P \models I_l(G)$$
$$\text{If } P \models I_l(G) \text{ then } P \models I_p(G)$$

Graph Separation: Examples



Example of global independence I (G): $\{A\} \perp \{G\} \mid \{D\}$

Example of pairwise independence I_p (G): $A \perp G \mid \{B,C,D,E,F\}$

Example of local independence I_l (G): $A \perp \{D, E,F,G\} \mid \{B,C\}$

Graphical Models: So far

So far we learned what a Markov Network is, what it means for a probability distribution to factor over a graph and how to extract information about independence of variables from the graph.

We next look at two equivalent representation of Markov Networks, one of which is amenable to inference queries (factor graphs) and another which is amenable to learning from data (e.g., log-linear form).

Log-Linear Models

Another representation of a Markov Network is that of a log-linear model

Here, we have a set of feature functions $f_1(\mathrm{D}_1),\dots, f_k(\mathrm{D}_k)$ where each D_i is a complete subgraph of G. An $f_i(\mathrm{D}_i)$ can return any value, i.e., can be negative

$$P(X_1,\dots,Xn) = \frac{1}{Z(X_1,\dots,Xn)} exp^{-\sum_{i=1}^{k}(w_i \times f_i(\mathrm{D}_i))}$$

$$Z(X_1, \dots, X_n) = \sum exp^{-\sum_{i=1}^{k}(w_i \times f_i(\mathrm{D}_i))}$$

Any Markov Network whose factors are positive can be converted to a log-linear model

Log-Linear Models to Markov Networks

Let $\varphi_i\,(\mathrm{D}_i) = exp^{-wi \times f_i(\mathrm{D}_i)}$

Then

$$exp^{-\sum_{i=1}^{k}(w_i \times f_i(\mathrm{D}_i))} =$$

$$exp^{(-w_1 \times f_1(\mathrm{D}_1) - w_2 \times f_2(\mathrm{D}_2) \ldots - wk \times f_k(\mathrm{D}_k))} =$$

$$exp^{-w_1 \times f_1(\mathrm{D}_1)} \times exp^{-w_2 \times f_2(\mathrm{D}_2)} \ldots \times exp^{(-wk \times f_k(\mathrm{D}_k)} =$$

$$\varphi_1\,(\mathrm{D}_1) \times \varphi_2\,(\mathrm{D}_2) \ldots \times \varphi_k\,(\mathrm{D}_k)$$

By substitution:

$$\mathrm{P}(\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_n) = \frac{\varphi_1\,(\mathrm{D}_1) \times \varphi_2\,(\mathrm{D}_2) \times \ldots \times \varphi_k\,(\mathrm{D}_k)}{\mathrm{Z}(\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_n)}$$

Log-Linear Representation: Benefits

$$P(X_1,\ldots,Xn) = \frac{1}{Z(X_1,\ldots,Xn)} exp^{-\sum_{i=1}^{k}(w_i \times f_i(\mathrm{D}_i))}$$

$$Z(X_1, \ldots, X_n) = \sum exp^{-\sum_{i=1}^{k}(w_i \times f_i(\mathrm{D}_i))}$$

Log-linear models have few benefits over factors. They:

1. Make certain relationships more explicit

2. Offer a much more compact representation for many distributions, especially for variables with large domains (e.g., names in JSNice)

3. They are useful for learning where we are given the feature functions and the learning phase simply figures out the weights.

Feature Function: Example

For example, suppose that for the factor φ_1 :

$$\varphi_1(\mathrm{A}, \mathrm{B}) = \begin{cases} 148.41 & \text{if } \mathrm{A} = \mathrm{B} \\ 1 & \text{otherwise} \end{cases}$$

If A and B range over m and n values respectively, we have $m \times n$ possible values to store in order to keep this factor.

Instead, we can use a single feature function $f_{\mathrm{A}\,=\,\mathrm{B}}(\mathrm{A}, \mathrm{B})$ where

$$f_1(\mathrm{A}, \mathrm{B}) = \begin{cases} 1 & \text{if } \mathrm{A} = \mathrm{B} \\ 0 & \text{else} \end{cases} \qquad \text{and} \qquad w_1 = -5$$

Factors vs. Features + Weights

φ_1 (A,B)

A	B	value
0	0	30
0	1	5
1	0	1
1	1	10

φ_2 (B,C)

B	C	value
0	0	100
0	1	1
1	0	1
1	1	100

φ_3 (C,D)

C	D	value
0	0	1
0	1	100
1	0	100
1	1	1

φ_4 (D,A)

D	A	value
0	0	100
0	1	1
1	0	1
1	1	100

Factors vs. Features + Weights

Recall $\varphi_i\,(\mathrm{D}_i) = exp^{\,-wi\,\times\, f_i(\mathrm{D}_i)}$

φ_1 (A,B)

A	B	value
0	0	30
0	1	5
1	0	1
1	1	10

Factors vs. Features + Weights

Recall $\varphi_i\ (\mathrm{D}_i) = exp^{-wi \times f_i(\mathrm{D}_i)}$

φ_1 (A,B)

A	B	value
0	0	30
0	1	5
1	0	1
1	1	10

$w_1 \times f_1(\mathrm{A},\mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A},\mathrm{B})$ = value

Factors vs. Features + Weights

Recall $\varphi_i\,(\mathrm{D}_i) = exp^{-wi \times f_i(\mathrm{D}_i)}$

φ_1 (A,B)

A	B	value
0	0	30
0	1	5
1	0	1
1	1	10

φ_2 (B,C)

B	C	value
0	0	100
0	1	1
1	0	1
1	1	100

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B})$ = value

Factors vs. Features + Weights

Recall $\varphi_i\,(\mathrm{D}_i) = exp^{\,-wi \,\times\, f_i(\mathrm{D}_i)}$

φ_1 (A,B)

A	B	value
0	0	30
0	1	5
1	0	1
1	1	10

φ_2 (B,C)

B	C	value
0	0	100
0	1	1
1	0	1
1	1	100

$w_1 \times f_1(\mathrm{A},\mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A},\mathrm{B})$ = value

$w_2 \times f_2(\mathrm{B},\mathrm{C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

w_2 = −4.61
$f_2(\mathrm{B},\mathrm{C})$ = 1 if B = C
0 else

Factors vs. Features + Weights

Recall $\varphi_i\,(\mathrm{D}_i) = exp^{\,-wi \times f_i(\mathrm{D}_i)}$

φ_1 (A,B)

A	B	value
0	0	30
0	1	5
1	0	1
1	1	10

φ_2 (B,C)

B	C	value
0	0	100
0	1	1
1	0	1
1	1	100

φ_3 (C,D)

C	D	value
0	0	1
0	1	100
1	0	100
1	1	1

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B})$ = value

$w_2 \times f_2(\mathrm{B}, \mathrm{C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

w_2 = −4.61
$f_2(\mathrm{B}, \mathrm{C})$ = 1 if B = C
0 else

Factors vs. Features + Weights

Recall $\varphi_i\,(\mathrm{D}_i) = exp^{-wi \times f_i(\mathrm{D}_i)}$

φ_1 (A,B)

A	B	value
0	0	30
0	1	5
1	0	1
1	1	10

φ_2 (B,C)

B	C	value
0	0	100
0	1	1
1	0	1
1	1	100

φ_3 (C,D)

C	D	value
0	0	1
0	1	100
1	0	100
1	1	1

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B})$ = value

$w_2 \times f_2(\mathrm{B}, \mathrm{C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

w_2 = −4.61
$f_2(\mathrm{B}, \mathrm{C})$ = 1 if B = C
0 else

$w_3 \times f_3(\mathrm{C}, \mathrm{D})$

C	D	value
0	0	0
0	1	-4.61
1	0	-4.61
1	1	0

w_3 = −4.61
$f_3(\mathrm{C}, \mathrm{D})$ = 1 if C ≠ D
0 else

Factors vs. Features + Weights

Recall $\varphi_i\,(\mathrm{D}_i) = exp^{\,-wi \times f_i(\mathrm{D}_i)}$

φ_1 (A,B)

A	B	value
0	0	30
0	1	5
1	0	1
1	1	10

φ_2 (B,C)

B	C	value
0	0	100
0	1	1
1	0	1
1	1	100

φ_3 (C,D)

C	D	value
0	0	1
0	1	100
1	0	100
1	1	1

φ_4 (D,A)

D	A	value
0	0	100
0	1	1
1	0	1
1	1	100

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B})$ = value

$w_2 \times f_2(\mathrm{B}, \mathrm{C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

w_2 = −4.61
$f_2(\mathrm{B}, \mathrm{C})$ = 1 if B = C
0 else

$w_3 \times f_3(\mathrm{C}, \mathrm{D})$

C	D	value
0	0	0
0	1	-4.61
1	0	-4.61
1	1	0

w_3 = −4.61
$f_3(\mathrm{C}, \mathrm{D})$ = 1 if C ≠ D
0 else

Factors vs. Features + Weights

Recall $\varphi_i\,(\mathrm{D}_i) = exp^{-wi \times f_i(\mathrm{D}_i)}$

φ_1 (A,B)

A	B	value
0	0	30
0	1	5
1	0	1
1	1	10

φ_2 (B,C)

B	C	value
0	0	100
0	1	1
1	0	1
1	1	100

φ_3 (C,D)

C	D	value
0	0	1
0	1	100
1	0	100
1	1	1

φ_4 (D,A)

D	A	value
0	0	100
0	1	1
1	0	1
1	1	100

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B})$ = value

$w_2 \times f_2(\mathrm{B}, \mathrm{C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

w_2 = −4.61
$f_2(\mathrm{B}, \mathrm{C})$ = 1 if B = C
0 else

$w_3 \times f_3(\mathrm{C}, \mathrm{D})$

C	D	value
0	0	0
0	1	-4.61
1	0	-4.61
1	1	0

w_3 = −4.61
$f_3(\mathrm{C}, \mathrm{D})$ = 1 if C ≠ D
0 else

$w_4 \times f_4(\mathrm{D}, \mathrm{A})$

D	A	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

w_4 = −4.61
$f_4(\mathrm{D}, \mathrm{A})$ = 1 if D = A
0 else

Indicator Feature Function

There is a special, restricted kind of a simple feature function which is particularly important in practice, called the indicator feature function

This function is one which takes the value of 1 for particular values of the arguments D_i and 0 for all other values.

We use the notation $f^{\mathrm{ab}}(\mathrm{A}, \mathrm{B})$ to denote that $f(\mathrm{A}, \mathrm{B})$ returns 1 when A = a and B = b and 0 otherwise.

Indicator functions can be directly extracted from data during learning (later)

Example: Using Indicator Function

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B})$ = value

$w_2 \times f_2(\mathrm{B}, \mathrm{C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

$w_2 = -4.61$
$f_2(\mathrm{B}, \mathrm{C}) = 1$ if B = C
0 else

$w_3 \times f_3(\mathrm{C}, \mathrm{D})$

C	D	value
0	0	0
0	1	-4.61
1	0	-4.61
1	1	0

$w_3 = -4.61$
$f_3(\mathrm{C}, \mathrm{D}) = 1$ if C $\neq$ D
0 else

$w_4 \times f_4(\mathrm{D}, \mathrm{A})$

D	A	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

$w_4 = -4.61$
$f_4(\mathrm{D}, \mathrm{A}) = 1$ if D = A
0 else

Example: Using Indicator Function

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B}) =$ value

Example: Using Indicator Function

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B}) =$ value

$f_1{}^{00}(\mathrm{A}, \mathrm{B}) = 1$ if
A = 0, B = 0

$f_1{}^{01}(\mathrm{A}, \mathrm{B}) = 1$ if
A = 0, B = 1

$f_1{}^{10}(\mathrm{A}, \mathrm{B}) = 1$ if
A = 1, B = 0

$f_1{}^{11}(\mathrm{A}, \mathrm{B}) = 1$ if
A = 1, B = 1

Example: Using Indicator Function

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B}) =$ value

$f_1{}^{00}(\mathrm{A}, \mathrm{B}) =$ 1 if
A = 0, B = 0

$f_1{}^{01}(\mathrm{A}, \mathrm{B}) =$ 1 if
A = 0, B = 1

$f_1{}^{10}(\mathrm{A}, \mathrm{B}) =$ 1 if
A = 1, B = 0

$f_1{}^{11}(\mathrm{A}, \mathrm{B}) =$ 1 if
A = 1, B = 1

$f_1(\mathrm{A}, \mathrm{B}) =$

$$
\begin{aligned}
&-3.4 \times f_1{}^{00}(\mathrm{A}, \mathrm{B}) + \\
&-1.61 \times f_1{}^{01}(\mathrm{A}, \mathrm{B}) + \\
&\quad 1 \times f_1{}^{10}(\mathrm{A}, \mathrm{B}) + \\
&-2.3 \times f_1{}^{11}(\mathrm{A}, \mathrm{B})
\end{aligned}
$$

Example: Using Indicator Function

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

$w_1 = 1$
$f_1(\mathrm{A}, \mathrm{B}) = \text{value}$

$f_1(\mathrm{A}, \mathrm{B}) =$

$$\begin{aligned}
&-3.4 \times f_1^{00}(\mathrm{A}, \mathrm{B}) + \\
&-1.61 \times f_1^{01}(\mathrm{A}, \mathrm{B}) + \\
&\quad 1 \times f_1^{10}(\mathrm{A}, \mathrm{B}) + \\
&-2.3 \times f_1^{11}(\mathrm{A}, \mathrm{B})
\end{aligned}$$

Example: Using Indicator Function

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B}) =$ value

$w_2 \times f_2(\mathrm{B}, \mathrm{C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

$w_2 = -4.61$
$f_2(\mathrm{B}, \mathrm{C}) =$ 1 if B = C
0 else

$f_1(\mathrm{A}, \mathrm{B}) =$

$$\begin{aligned}
&\text{-3.4} \times f_1{}^{00}(\mathrm{A}, \mathrm{B}) + \\
&\text{-1.61} \times f_1{}^{01}(\mathrm{A}, \mathrm{B}) + \\
&\quad 1 \times f_1{}^{10}(\mathrm{A}, \mathrm{B}) + \\
&\text{-2.3} \times f_1{}^{11}(\mathrm{A}, \mathrm{B})
\end{aligned}$$

Example: Using Indicator Function

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B})$ = value

$w_2 \times f_2(\mathrm{B}, \mathrm{C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

w_2 = −4.61
$f_2(\mathrm{B}, \mathrm{C})$ = 1 if B = C
0 else

$f_1(\mathrm{A}, \mathrm{B}) =$

$$
\begin{aligned}
&-3.4 \times f_1^{00}(\mathrm{A}, \mathrm{B}) + \\
&-1.61 \times f_1^{01}(\mathrm{A}, \mathrm{B}) + \\
&\quad 1 \times f_1^{10}(\mathrm{A}, \mathrm{B}) + \\
&-2.3 \times f_1^{11}(\mathrm{A}, \mathrm{B})
\end{aligned}
$$

$f_2^{00}(\mathrm{B}, \mathrm{C})$ = 1 if
B = 0, C = 0

$f_2^{01}(\mathrm{B}, \mathrm{C})$ = 1 if
B = 0, C = 1

$f_2^{10}(\mathrm{B}, \mathrm{C})$ = 1 if
B = 1, C = 0

$f_2^{11}(\mathrm{B}, \mathrm{C})$ = 1 if
B = 1, C = 1

Example: Using Indicator Function

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B})$ = value

$w_2 \times f_2(\mathrm{B}, \mathrm{C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

w_2 = −4.61
$f_2(\mathrm{B}, \mathrm{C})$ = 1 if B = C
0 else

$f_1(\mathrm{A}, \mathrm{B}) =$

-3.4 $\times f_1^{00}(\mathrm{A}, \mathrm{B}) +$
-1.61 $\times f_1^{01}(\mathrm{A}, \mathrm{B}) +$
1 $\times f_1^{10}(\mathrm{A}, \mathrm{B}) +$
-2.3 $\times f_1^{11}(\mathrm{A}, \mathrm{B})$

$f_2^{00}(\mathrm{B}, \mathrm{C})$ = 1 if
B = 0, C = 0

$f_2^{01}(\mathrm{B}, \mathrm{C})$ = 1 if
B = 0, C = 1

$f_2^{10}(\mathrm{B}, \mathrm{C})$ = 1 if
B = 1, C = 0

$f_2^{11}(\mathrm{B}, \mathrm{C})$ = 1 if
B = 1, C = 1

$f_2(\mathrm{B}, \mathrm{C}) =$
$f_2^{00}(\mathrm{B}, \mathrm{C}) + f_2^{11}(\mathrm{B}, \mathrm{C})$

Example: Using Indicator Function

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B}) =$ value

$w_2 \times f_2(\mathrm{B}, \mathrm{C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

w_2 = −4.61
$f_2(\mathrm{B}, \mathrm{C}) =$ 1 if B = C
0 else

$f_1(\mathrm{A}, \mathrm{B}) =$

-3.4 $\times f_1{}^{00}(\mathrm{A}, \mathrm{B})$ +
-1.61$\times f_1{}^{01}(\mathrm{A}, \mathrm{B})$ +
1$\times f_1{}^{10}(\mathrm{A}, \mathrm{B})$ +
-2.3 $\times f_1{}^{11}(\mathrm{A}, \mathrm{B})$

$f_2(\mathrm{B}, \mathrm{C}) =$
$f_2{}^{00}(\mathrm{B}, \mathrm{C}) + f_2{}^{11}(\mathrm{B}, \mathrm{C})$

Example: Using Indicator Function

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

$w_1 = 1$
$f_1(\mathrm{A}, \mathrm{B}) =$ value

$$f_1(\mathrm{A}, \mathrm{B}) =$$

$$\text{-3.4} \times f_1^{00}(\mathrm{A}, \mathrm{B}) + \text{-1.61} \times f_1^{01}(\mathrm{A}, \mathrm{B}) + 1 \times f_1^{10}(\mathrm{A}, \mathrm{B}) + \text{-2.3} \times f_1^{11}(\mathrm{A}, \mathrm{B})$$

$w_2 \times f_2(\mathrm{B}, \mathrm{C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

$w_2 = -4.61$
$f_2(\mathrm{B}, \mathrm{C}) =$ 1 if B = C
0 else

$$f_2(\mathrm{B}, \mathrm{C}) = f_2^{00}(\mathrm{B}, \mathrm{C}) + f_2^{11}(\mathrm{B}, \mathrm{C})$$

Rest are Similar

Example: Using Indicator Function

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

$w_1 = 1$
$f_1(\mathrm{A}, \mathrm{B}) = \text{value}$

$f_1(\mathrm{A}, \mathrm{B}) =$

$$-3.4 \times f_1^{00}(\mathrm{A}, \mathrm{B}) + \\ -1.61 \times f_1^{01}(\mathrm{A}, \mathrm{B}) + \\ 1 \times f_1^{10}(\mathrm{A}, \mathrm{B}) + \\ -2.3 \times f_1^{11}(\mathrm{A}, \mathrm{B})$$

$w_2 \times f_2(\mathrm{B}, \mathrm{C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

$w_2 = -4.61$
$f_2(\mathrm{B}, \mathrm{C}) = 1$ if B = C
0 else

$f_2(\mathrm{B}, \mathrm{C}) = f_2^{00}(\mathrm{B}, \mathrm{C}) + f_2^{11}(\mathrm{B}, \mathrm{C})$

$w_3 \times f_3(\mathrm{C}, \mathrm{D})$

C	D	value
0	0	0
0	1	-4.61
1	0	-4.61
1	1	0

$w_3 = -4.61$
$f_3(\mathrm{C}, \mathrm{D}) = 1$ if C $\neq$ D
0 else

$f_3^{00}(\mathrm{C}, \mathrm{D}) = 1$ if C = 0, D = 0

$f_3^{01}(\mathrm{C}, \mathrm{D}) = 1$ if C = 0, D = 1

$f_3^{10}(\mathrm{C}, \mathrm{D}) = 1$ if C = 1, D = 0

$f_3^{11}(\mathrm{C}, \mathrm{D}) = 1$ if C = 1, D = 1

$w_4 \times f_4(\mathrm{D}, \mathrm{A})$

D	A	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

$w_4 = -4.61$
$f_4(\mathrm{D}, \mathrm{A}) = 1$ if D = A
0 else

$f_4^{00}(\mathrm{D}, \mathrm{A}) = 1$ if D = 0, A = 0

$f_4^{01}(\mathrm{C}, \mathrm{D}) = 1$ if D = 0, A = 1

$f_4^{10}(\mathrm{C}, \mathrm{D}) = 1$ if D = 1, A = 0

$f_4^{11}(\mathrm{C}, \mathrm{D}) = 1$ if D = 1, A = 1

Example: Using Indicator Function

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B})$ = value

$f_1(\mathrm{A}, \mathrm{B}) =$

-3.4 $\times f_1^{00}(\mathrm{A}, \mathrm{B}) +$
-1.61$\times f_1^{01}(\mathrm{A}, \mathrm{B}) +$
1$\times f_1^{10}(\mathrm{A}, \mathrm{B}) +$
-2.3 $\times f_1^{11}(\mathrm{A}, \mathrm{B})$

$w_2 \times f_2(\mathrm{B}, \mathrm{C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

$w_2 = -4.61$
$f_2(\mathrm{B}, \mathrm{C}) = 1$ if B = C
0 else

$f_2(\mathrm{B}, \mathrm{C}) =$
$f_2^{00}(\mathrm{B}, \mathrm{C}) + f_2^{11}(\mathrm{B}, \mathrm{C})$

$w_3 \times f_3(\mathrm{C}, \mathrm{D})$

C	D	value
0	0	0
0	1	-4.61
1	0	-4.61
1	1	0

$w_3 = -4.61$
$f_3(\mathrm{C}, \mathrm{D}) = 1$ if C $\neq$ D
0 else

$f_3(\mathrm{C}, \mathrm{D}) =$
$f_3^{01}(\mathrm{C}, \mathrm{D}) + f_3^{10}(\mathrm{C}, \mathrm{D})$

$w_4 \times f_4(\mathrm{D}, \mathrm{A})$

D	A	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

$w_4 = -4.61$
$f_4(\mathrm{D}, \mathrm{A}) = 1$ if D = A
0 else

$f_4(\mathrm{D}, \mathrm{A}) =$
$f_4^{00}(\mathrm{D}, \mathrm{A}) + f_4^{11}(\mathrm{D}, \mathrm{A}))$

Example: Using Indicator Function

$w_1 \times f_1(\mathrm{A}, \mathrm{B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A}, \mathrm{B})$ = value

$f_1(\mathrm{A}, \mathrm{B}) =$

$$-3.4 \times f_1^{00}(\mathrm{A}, \mathrm{B}) + -1.61 \times f_1^{01}(\mathrm{A}, \mathrm{B}) + 1 \times f_1^{10}(\mathrm{A}, \mathrm{B}) + -2.3 \times f_1^{11}(\mathrm{A}, \mathrm{B})$$

$w_2 \times f_2(\mathrm{B}, \mathrm{C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

w_2 = −4.61
$f_2(\mathrm{B}, \mathrm{C})$ = 1 if B = C
0 else

$f_2(\mathrm{B}, \mathrm{C}) = f_2^{00}(\mathrm{B}, \mathrm{C}) + f_2^{11}(\mathrm{B}, \mathrm{C})$

$w_3 \times f_3(\mathrm{C}, \mathrm{D})$

C	D	value
0	0	0
0	1	-4.61
1	0	-4.61
1	1	0

w_3 = −4.61
$f_3(\mathrm{C}, \mathrm{D})$ = 1 if C ≠ D
0 else

$f_3(\mathrm{C}, \mathrm{D}) = f_3^{01}(\mathrm{C}, \mathrm{D}) + f_3^{10}(\mathrm{C}, \mathrm{D})$

$w_4 \times f_4(\mathrm{D}, \mathrm{A})$

D	A	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

w_4 = −4.61
$f_4(\mathrm{D}, \mathrm{A})$ = 1 if D = A
0 else

$f_4(\mathrm{D}, \mathrm{A}) = f_4^{00}(\mathrm{D}, \mathrm{A}) + f_4^{11}(\mathrm{D}, \mathrm{A}))$

$$\mathrm{P}(\mathrm{X}_1,\ldots,\mathrm{Xn}) = \frac{1}{\mathrm{Z}(\mathrm{X}_1,\ldots,\mathrm{Xn})} exp^{-\sum_{i=1}^{k}(w_i \times f_i(\mathrm{D}_i))}$$

Example: Using Indicator Function

$w_1 \times f_1(\mathrm{A,B})$

A	B	value
0	0	-3.4
0	1	-1.61
1	0	0
1	1	-2.3

w_1 = 1
$f_1(\mathrm{A,B})$ = value

$w_2 \times f_2(\mathrm{B,C})$

B	C	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

$w_2 = -4.61$
$f_2(\mathrm{B,C})$ = 1 if B = C
0 else

$w_3 \times f_3(\mathrm{C,D})$

C	D	value
0	0	0
0	1	-4.61
1	0	-4.61
1	1	0

$w_3 = -4.61$
$f_3(\mathrm{C,D})$ = 1 if C ≠ D
0 else

$w_4 \times f_4(\mathrm{D,A})$

D	A	value
0	0	-4.61
0	1	0
1	0	0
1	1	-4.61

$w_4 = -4.61$
$f_4(\mathrm{D,A})$ = 1 if D = A
0 else

$f_1(\mathrm{A,B}) =$

$$\begin{aligned} &-3.4 \times f_1^{00}(\mathrm{A,B}) + \\ &-1.61 \times f_1^{01}(\mathrm{A,B}) + \\ &\quad 1 \times f_1^{10}(\mathrm{A,B}) + \\ &-2.3 \times f_1^{11}(\mathrm{A,B}) \end{aligned}$$

$f_2(\mathrm{B,C}) =$
$f_2^{00}(\mathrm{B,C}) + f_2^{11}(\mathrm{B,C})$

$f_3(\mathrm{C,D}) =$
$f_3^{01}(\mathrm{C,D}) + f_3^{10}(\mathrm{C,D})$

$f_4(\mathrm{D,A}) =$
$f_4^{00}(\mathrm{D,A}) + f_4^{11}(\mathrm{D,A}))$

$$\mathrm{P(A, B, C, D)} = \frac{exp^{-\mathrm{SP}}}{\mathrm{Z(A,B,C,D)}}$$

$$\mathrm{SP} = -3.4 \times f_1^{00}(\mathrm{A,B}) + -1.61 \times f_1^{01}(\mathrm{A,B}) + 1 \times f_1^{10}(\mathrm{A,B}) + -2.3 \times f_1^{11}(\mathrm{A,B}) + -4.61 \times f_2^{00}(\mathrm{B,C}) + -4.61 \times f_2^{11}(\mathrm{B,C}) + -4.61 \times f_3^{01}(\mathrm{C,D}) + -4.61 \times f_3^{10}(\mathrm{C,D}) + -4.61 \times f_4^{00}(\mathrm{D,A}) + -4.61 \times f_4^{11}(\mathrm{D,A}))$$

Graphical Models: So far

In the last segment we learned about log-linear models and indicator functions.

Log-linear models are important because they allow to capture factors more concisely, the indicator functions can be directly extracted from data, and the weights can be learned (to fit the optimization objective, discussed later).

So far we have been discussing the joint probability distribution $P(X_1,\ldots,Xn)$. We next focus on conditional distributions.

Conditional Random Field

[J. Lafferty, A. McCallum, F. Pereira, ICML 2001]

Let $X \cup Y$ be a set of random variables where $X = \{X_1,\dots, X_n\}$, $Y = \{Y_1,\dots, Y_n\}$.
Here, X are the observed variables and Y are the target variables.
Let $D_1, D_2, \dots, D_m \subseteq X \cup Y$ where $D_i \nsubseteq X$

A Conditional Random Field is defined as:

$$P(Y_1,\dots,Y_n \mid X_1,\dots, X_n) = \frac{\varphi_1(D_1) \times \varphi_2(D_2) \times \dots \times \varphi_m(D_m)}{Z(X_1, \dots, X_n)}$$

$$Z(X_1,\dots, X_n) = \sum_{Y} \varphi_1(D_1) \times \varphi_2(D_2) \times \dots \times \varphi_m(D_m)$$

Note that the Z function ranges over variables in X only.

Conditional Random Field

[J. Lafferty, A. McCallum, F. Pereira, ICML 2001]

Key advantage: avoids encoding distribution over the variables X. Thus, we can use many observed variables with complex dependencies without needing to model any joint distributions over them.

$$P(Y_1,\ldots,Y_n \mid X_1,\ldots,X_n) = \frac{\varphi_1(D_1) \times \varphi_2(D_2) \times \ldots \times \varphi_m(D_m)}{Z(X_1,\ldots,X_n)}$$

$$Z(X_1,\ldots,X_n) = \sum_{Y} \varphi_1(D_1) \times \varphi_2(D_2) \times \ldots \times \varphi_m(D_m)$$

Conditional Random Field: Example

Text Analytics

Problem Statement:

Given a sentence, label each word with whether it is a person or a location

Conditional Random Field: Example

Text Analytics

Problem Statement:

Given a sentence, label each word with whether it is a person or a location

5 Labels: B-per, I-per, B-loc, I-loc, N

Conditional Random Field: Example

Text Analytics

Problem Statement:

Given a sentence, label each word with whether it is a person or a location

5 Labels: B-per, I-per, B-loc, I-loc, N

Given:

Mr. Smith came today to Portland

Conditional Random Field: Example

Text Analytics

Problem Statement:

Given a sentence, label each word with whether it is a person or a location

5 Labels: B-per, I-per, B-loc, I-loc, N

Predict:	B-per	I-per	N	N	N	B-loc
Given:	Mr.	Smith	came	today	to	Portland

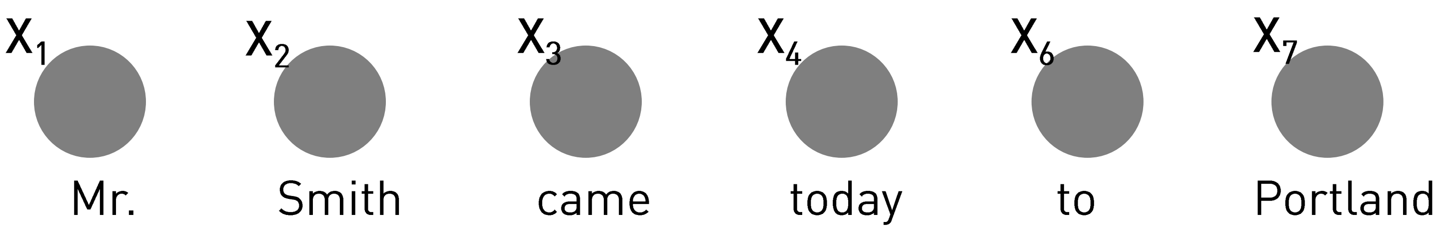
Conditional Random Field: Example

Text Analytics

<u>Problem Statement:</u>

Given a sentence, label each word with whether it is a person or a location

5 Labels: B-per, I-per, B-loc, I-loc, N



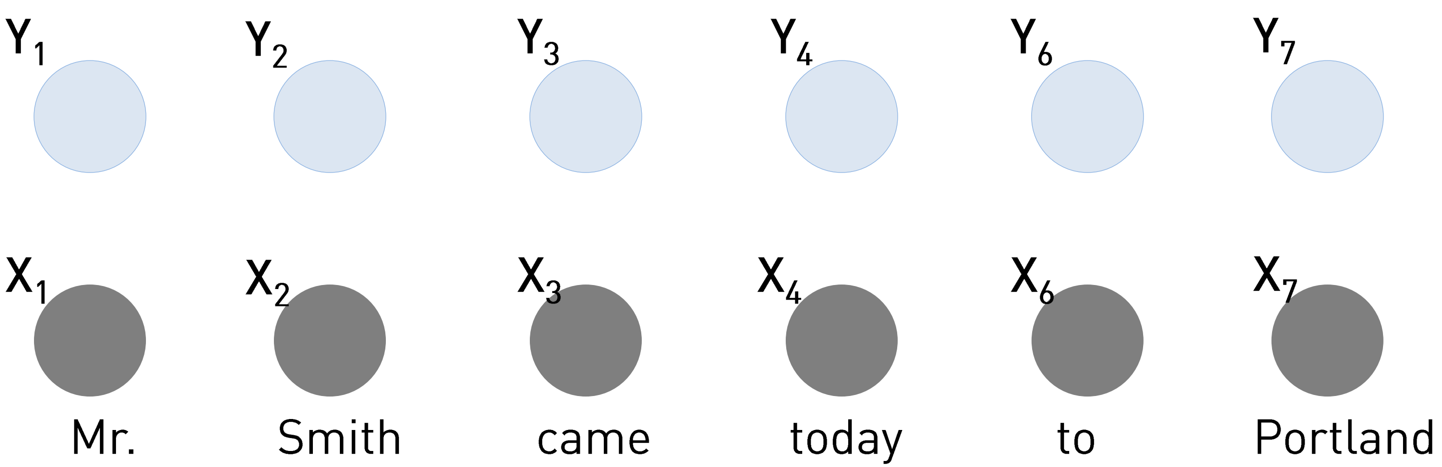
Conditional Random Field: Example

Text Analytics

<u>Problem Statement:</u>

Given a sentence, label each word with whether it is a person or a location

5 Labels: B-per, I-per, B-loc, I-loc, N



Conditional Random Field: Example

Text Analytics

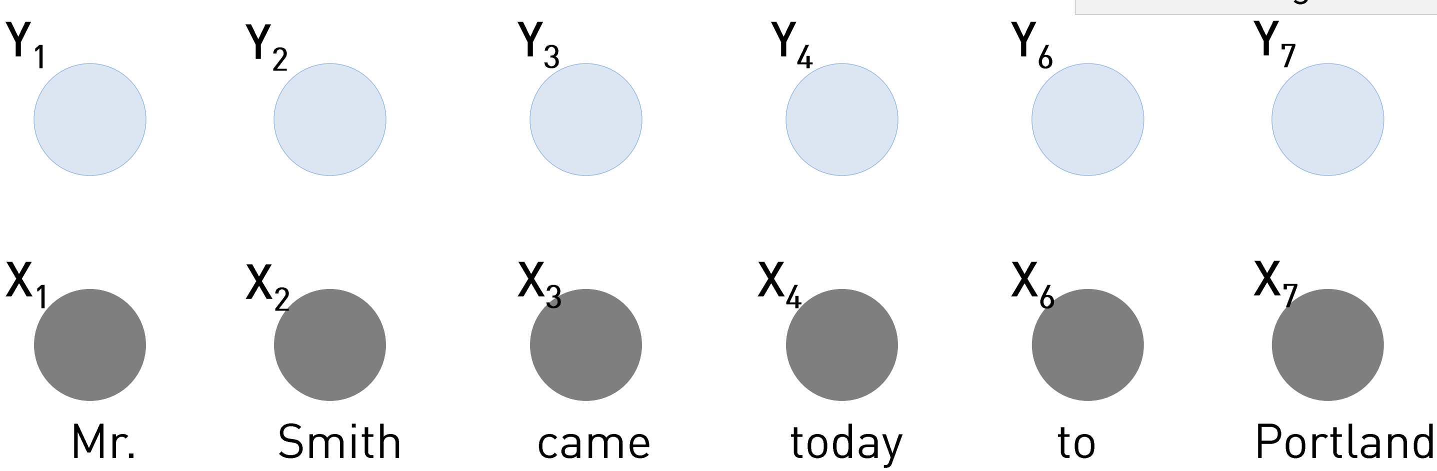
<u>Problem Statement:</u>
Given a sentence, label each word with whether it is a person or a location

5 Labels: B-per, I-per, B-loc, I-loc, N

Captures relationship between neighboring predictions

Two kinds of factors for each word X_t $\varphi^1_t(Y_t, Y_{t+1})$ $\varphi^2_t(Y_t, X_1, \ldots, X_t)$

Relationship between prediction and the neighbors of the word



Conditional Random Field: Example

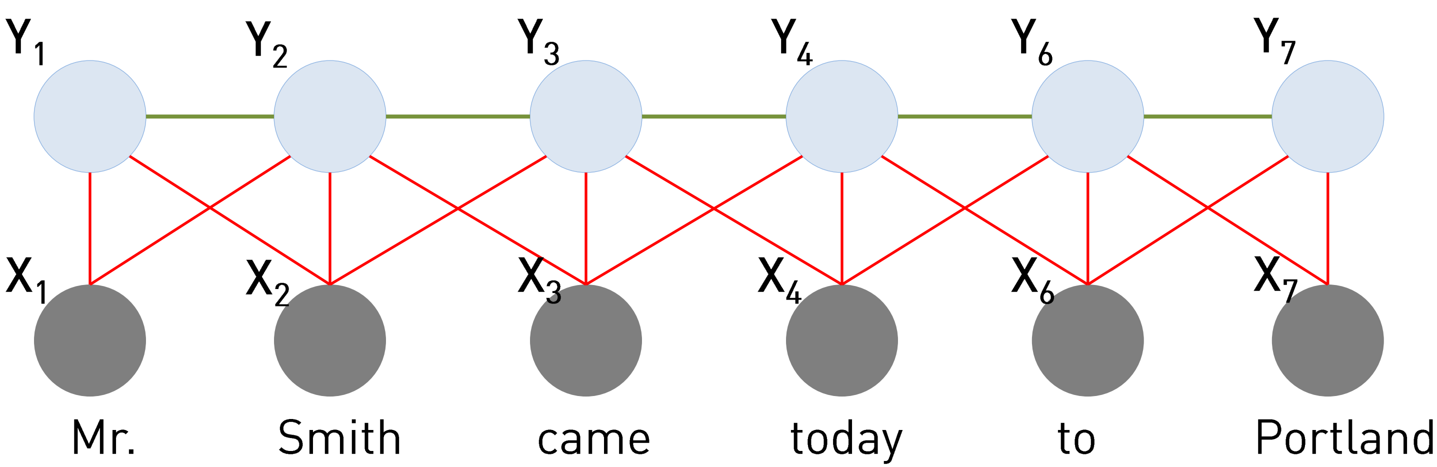
Text Analytics

Problem Statement:

Given a sentence, label each word with whether it is a person or a location

5 Labels: B-per, I-per, B-loc, I-loc, N

Two kinds of factors for each word X_t $\varphi^1_t(Y_t, Y_{t+1})$ $\varphi^2_t(Y_t, X_1, \ldots, X_t)$



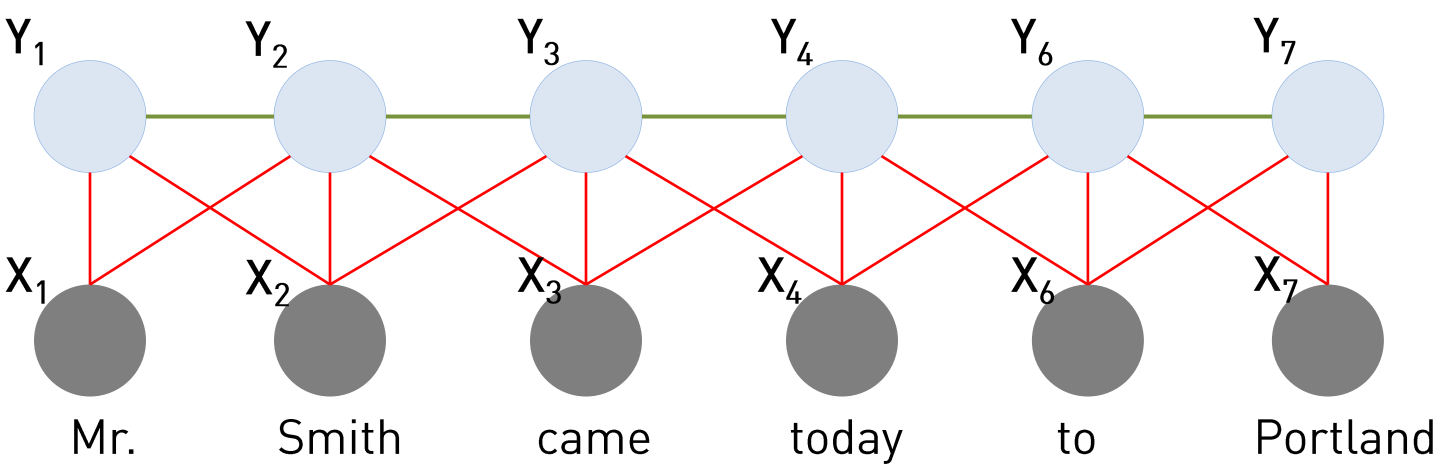
Conditional Random Field: Example

Text Analytics

<u>Problem Statement:</u>

Given a sentence, label each word with whether it is a person or a location

$$P(Y_1,\ldots,Y_7 \mid X_1,\ldots,X_7) = \frac{\varphi^1_1(Y_1, Y_2) \times \ldots \times \varphi^7_6(Y_6, Y_7) \times \varphi^2_1(Y_1,X_1,X_2) \times \ldots \times \varphi^2_7(Y_7,X_6,X_7)}{\sum_{Y_1,Y_2,Y_3,Y_4,Y_5,Y_6,Y_7} \varphi^1_1(Y_1, Y_2) \times \ldots \times \varphi^7_6(Y_6, Y_7) \times \varphi^2_1(Y_1,X_1,X_2) \times \ldots \times \varphi^2_7(Y_7,X_6,X_7)}$$



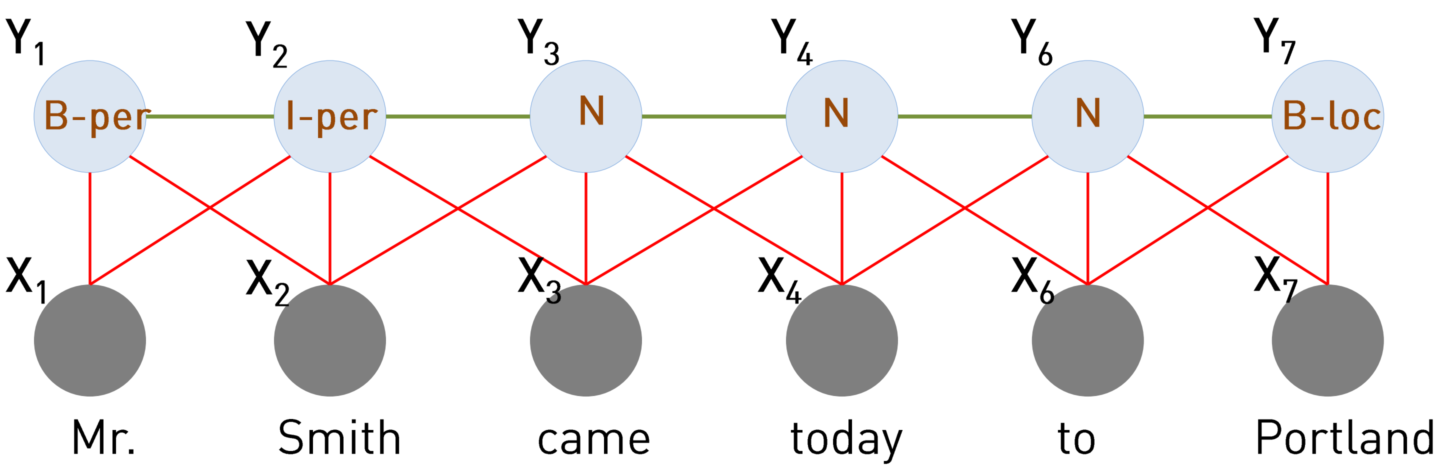
Conditional Random Field: Example

Text Analytics

Problem Statement:

Given a sentence, label each word with whether it is a person or a location

$$P(Y_1,\ldots,Y_7 \mid X_1,\ldots, X_7) = \frac{\varphi^1_1(Y_1, Y_2) \times \ldots \times \varphi^7_6(Y_6, Y_7) \times \varphi^2_1(Y_1,X_1,X_2) \times \ldots \times \varphi^2_7(Y_7,X_6,X_7)}{\sum_{Y_1,Y_2,Y_3,Y_4,Y_5,Y_6,Y_7} \varphi^1_1(Y_1, Y_2) \times \ldots \times \varphi^7_6(Y_6, Y_7) \times \varphi^2_1(Y_1,X_1,X_2) \times \ldots \times \varphi^2_7(Y_7,X_6,X_7)}$$



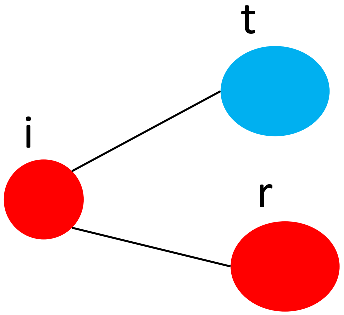
Conditional Random Field: Log-Linear Form

$$P(y \mid x) = \frac{1}{Z(x)} \exp(\boldsymbol{w}^T \boldsymbol{f}(y, x))$$

CRF: Example

$$P(y \mid x) = \frac{1}{Z(x)} \exp(\boldsymbol{w}^T \boldsymbol{f}(y, x))$$

Example: Let $y = i, r$ and $x = t$

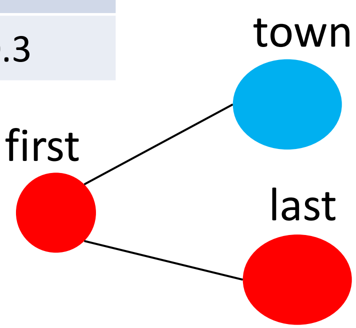


CRF: Example

$$P(\color{red}{y} \mid \color{cyan}{x}) = \frac{1}{Z(\color{cyan}{x})} \exp(\boldsymbol{w}^T \boldsymbol{f}(\color{red}{y}, \color{cyan}{x}))$$

Example: Let $\color{red}{y}$ = first, last and $\color{cyan}{x}$ = town

	first	town	w
f_1	Pencho	Portland	0.1
f_2	James	Portland	0.3



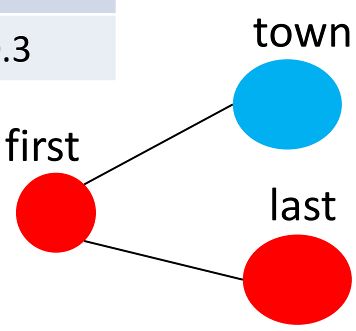
	first	last	w
f_3	Pencho	Smith	0.7
f_4	James	Chandra	0.4

CRF: Example

$$P(y \mid x) = \frac{1}{Z(x)} \exp(\boldsymbol{w}^T \boldsymbol{f}(y, x))$$

Example: Let y = first, last and x = town

	first	town	w
f_1	Pencho	Portland	0.1
f_2	James	Portland	0.3



	first	last	w
f_3	Pencho	Smith	0.7
f_4	James	Chandra	0.4

$$P(\text{first}, \text{last} \mid \text{town}) = \frac{\exp\,(\,0.1 * f_1 + 0.3 * f_2 + 0.7 * f_3 + 0.4 * f_4)}{Z(\text{town})}$$

Tutorial Outline

- ~~Motivation~~
  - ~~Potential applications~~

- ~~Statistical language models~~
  - ~~N-gram and Recurrent Networks, Smoothing~~
  - ~~Application: code completion~~

- Graphical Models
  - ~~Markov Networks, Conditional Random Fields~~
  - Inference in Markov Networks
  - Learning in Markov Networks
  - Application: predicting names and types

- Hands-on session with Nice2Predict

Queries: MAP vs. Max Marginals

MAP Inference:

$$(y_1, \dots, yn)^{\mathrm{MAP}} = \underset{y_1, \dots, yn}{\operatorname{argmax}} P(y_1, \dots, y_n)$$

VS.

Max Marginals:

$$(y_1, \dots, yn)^{\mathrm{ML}} = (y_1{}^{\mathrm{ML}}, \dots, yn^{\mathrm{ML}})$$

$$y_1{}^{\mathrm{ML}} = \underset{y_1}{\operatorname{argmax}} P(y_1) \quad \dots\dots \quad y_{\mathrm{n}}{}^{\mathrm{ML}} = \underset{y_n}{\operatorname{argmax}} P(y_n)$$

Consider the following probability distribution:

A	B	val
0	0	0.2
0	1	0.3
1	0	0.15
1	1	0.35

Queries: MAP vs. Max Marginals

MAP Inference: $(y_1, \ldots, yn)^{\mathrm{MAP}} = \underset{y_1, \ldots, yn}{\operatorname{argmax}} P(y_1, \ldots, y_n)$

VS.

Max Marginals: $(y_1, \ldots, yn)^{\mathrm{ML}} = (y_1^{\mathrm{ML}}, \ldots, yn^{\mathrm{ML}})$

$y_1^{\mathrm{ML}} = \underset{y_1}{\operatorname{argmax}} P(y_1)$ …… $y_n^{\mathrm{ML}} = \underset{y_n}{\operatorname{argmax}} P(y_n)$

Consider the following probability distribution:

$(1,1)^{\mathrm{MAP}}$

$\underset{A, B}{\operatorname{argmax}} P(A, B)$

$(0,1)^{\mathrm{ML}}$

$\underset{A}{\operatorname{argmax}} P(A)$ $\underset{B}{\operatorname{argmax}} P(B)$

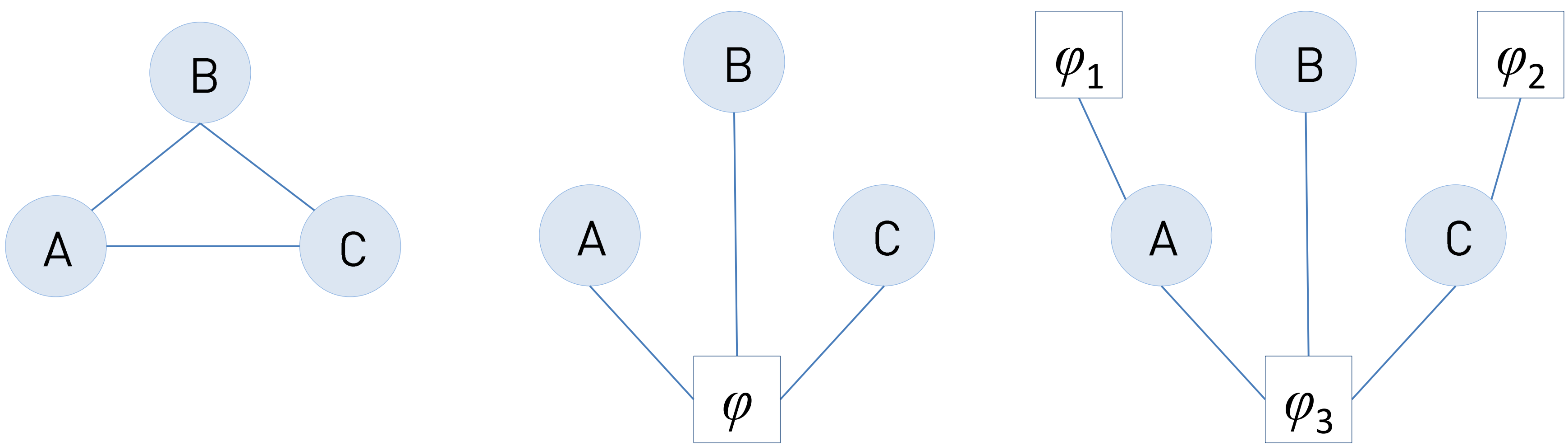
A	B	val
0	0	0.2
0	1	0.3
1	0	0.15
1	1	0.35

Factor Graph Representation

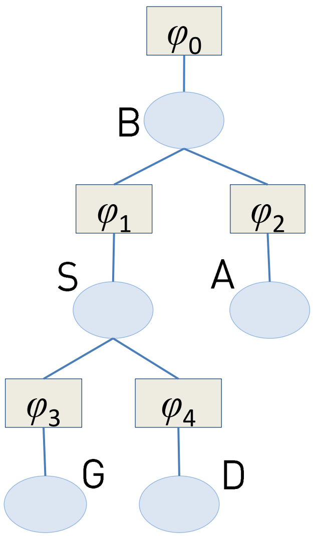
A third representation of a Markov Network, which makes the factors explicit in the graph is the factor graph. Here, factors are nodes in the graph. Edges only exist between factor nodes and variable nodes.

Factor graphs are often useful when performing inference.

(a) Markov Network (b) Factor Graph I (c) Factor Graph II



Example: Compute $P(B=0 \mid G = 1, D = 0, A = 1)$



φ_0 (B)

B	val
0	0.99
1	0.01

φ_1 (S,B)

S	B	val
0	0	0.5
0	1	0.1
1	0	0.5
1	1	0.9

φ_2 (A,B)

A	B	val
0	0	0.99
0	1	0.1
1	0	0.01
1	1	0.9

φ_3 (G,S)

G	S	val
0	0	0.7
0	1	0.4
1	0	0.3
1	1	0.6

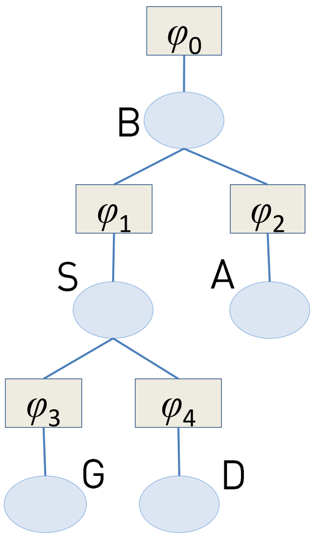
φ_4 (D,S)

D	S	val
0	0	0.8
0	1	0.3
1	0	0.2
1	1	0.7

$$P(B, S \mid G, D, A) = \frac{\varphi_0(B) \times \varphi_1(S, B) \times \varphi_2(A, B) \times \varphi_3(G, S) \times \varphi_4(D, S)}{Z(G, D, A)}$$

$$Z(G, D, A) = \sum_{B,S} \varphi_0(B) \times \varphi_1(S, B) \times \varphi_2(A, B) \times \varphi_3(G, S) \times \varphi_4(D, S)$$

Example: Compute P(B=0 | G = 1, D = 0, A = 1)



φ_0 (B)

B	val
0	0.99
1	0.01

φ_1 (S,B)

S	B	val
0	0	0.5
0	1	0.1
1	0	0.5
1	1	0.9

φ_2 (A,B)

A	B	val
0	0	0.99
0	1	0.1
1	0	0.01
1	1	0.9

φ_3 (G,S)

G	S	val
0	0	0.7
0	1	0.4
1	0	0.3
1	1	0.6

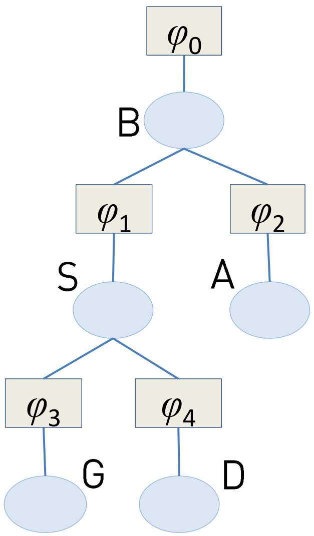
φ_4 (D,S)

D	S	val
0	0	0.8
0	1	0.3
1	0	0.2
1	1	0.7

$$P(B=0 \mid G, D, A) = \sum_S P(B=0, S \mid G, D, A) = P(B, S=0 \mid G, D, A) + P(B, S=1 \mid G, D, A)$$

$$P(B=0 \mid G=1, D=0, A=1) = P(B=0, S = 0 \mid G=1, D=0, A=1) + P(B=0, S=1 \mid G=1, D=0, A=1)$$

Example: Compute P(B=0 | G = 1, D = 0, A = 1)



φ_0 (B)

B	val
0	0.99
1	0.01

φ_1 (S,B)

S	B	val
0	0	0.5
0	1	0.1
1	0	0.5
1	1	0.9

φ_2 (A,B)

A	B	val
0	0	0.99
0	1	0.1
1	0	0.01
1	1	0.9

φ_3 (G,S)

G	S	val
0	0	0.7
0	1	0.4
1	0	0.3
1	1	0.6

φ_4 (D,S)

D	S	val
0	0	0.8
0	1	0.3
1	0	0.2
1	1	0.7

Z(G=1, D=0, A=1) =

$\varphi_0(0) \times \varphi_1(0, 0) \times \varphi_2(1, 0) \times \varphi_3(1, 0) \times \varphi_4(0, 0) + \quad \varphi_0(1) \times \varphi_1(0, 1) \times \varphi_2(1, 1) \times \varphi_3(1, 0) \times \varphi_4(0, 0) +$

$\varphi_0(0) \times \varphi_1(1, 0) \times \varphi_2(1, 0) \times \varphi_3(1, 1) \times \varphi_4(0, 1) + \quad \varphi_0(1) \times \varphi_1(1, 1) \times \varphi_2(1, 1) \times \varphi_3(1, 1) \times \varphi_4(0, 1) +$

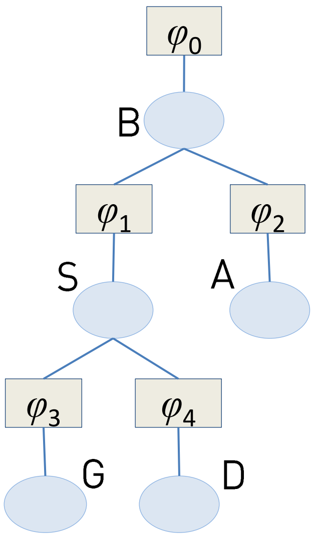
=

$0.99 \times 0.5 \times 0.01 \times 0.3 \times 0.8 + 0.01 \times 0.1 \times 0.9 \times 0.3 \times 0.8 +$

$0.99 \times 0.5 \times 0.01 \times 0.6 \times 0.3 + 0.01 \times 0.9 \times 0.9 \times 0.6 \times 0.7$

$= 0.001188 + 0.000216 + 0.000891 + 0.003402 = 0.005697$

Example: Compute P(B=0 | G = 1, D = 0, A = 1)



φ_0 (B)

B	val
0	0.99
1	0.01

φ_1 (S,B)

S	B	val
0	0	0.5
0	1	0.1
1	0	0.5
1	1	0.9

φ_2 (A,B)

A	B	val
0	0	0.99
0	1	0.1
1	0	0.01
1	1	0.9

φ_3 (G,S)

G	S	val
0	0	0.7
0	1	0.4
1	0	0.3
1	1	0.6

φ_4 (D,S)

D	S	val
0	0	0.8
0	1	0.3
1	0	0.2
1	1	0.7

P(B=0, S = 0 | G=1, D=0, A=1)
$= \varphi_0 (B) \times \varphi_1 (S, B) \times \varphi_2 (A, B) \times \varphi_3 (G, S) \times \varphi_4 (D, S)/Z =$
$= \varphi_0 (0) \times \varphi_1 (0, 0) \times \varphi_2 (1, 0) \times \varphi_3 (1, 0) \times \varphi_4 (0, 0)/Z$
$= 0.99 \times 0.5 \times 0.01 \times 0.3 \times 0.8)/Z = 0.001188/0.005697$

$+$

P(B=0, S = 1 | G=1, D=0, A=1)
$= \varphi_0 (B) \times \varphi_1 (S, B) \times \varphi_2 (A, B) \times \varphi_3 (G, S) \times \varphi_4 (D, S)/Z =$
$= \varphi_0 (0) \times \varphi_1 (1, 0) \times \varphi_2 (1, 0) \times \varphi_3 (1, 1) \times \varphi_4 (0, 1)/Z$
$= 0.99 \times 0.5 \times 0.01 \times 0.6 \times 0.3)/Z = 0.000891/0.005697$

0.002079/0.005697

Belief Propagation

An inference algorithm for computing queries on distributions represented as factor graphs.

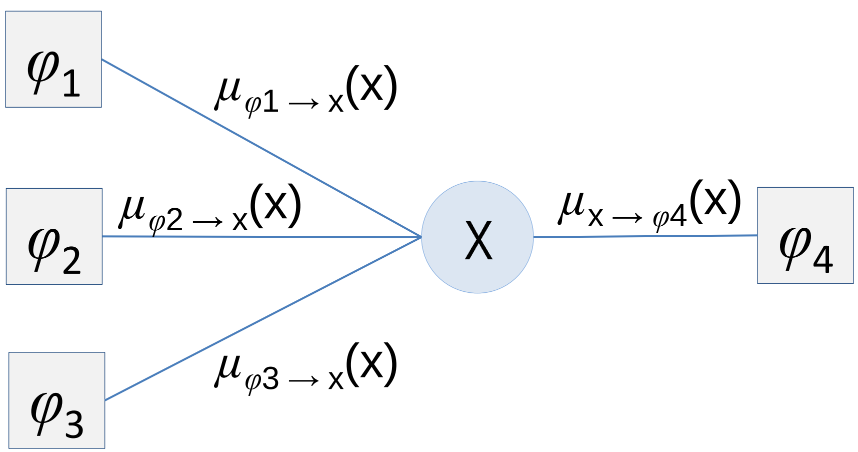
The algorithm is exact if the graph is in the shape of a tree. If the graph contains loops, the algorithm is referred to as loopy belief propagation and in this case, there are no guarantees on exactness. Nonetheless, the algorithm is often used for loopy graphs as well.

Algorithm is based on passing two kinds of messages on the factor graph.

Belief Propagation: Message Kinds

Variable-to-Factor

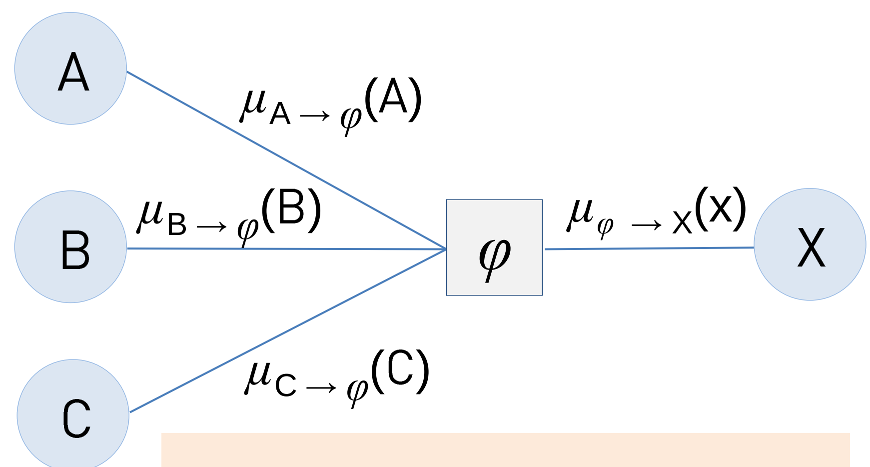
$\mu_{V \to \varphi}$: Val $\longrightarrow$ MsgVal



$$\mu_{X \to \varphi 4}(X) = \mu_{\varphi 1 \to X}(X) \times \mu_{\varphi 2 \to X}(X) \times \mu_{\varphi 3 \to X}(X)$$

Factor-to-Variable

$\mu_{\varphi \to V}$: Val $\longrightarrow$ (MsgVal, $\rho(V \times \mathrm{Val})$)



$$\mu_{\varphi \to X}(X) = \sum_{A,B,C} \Big(\varphi(A,B,C,X) \times \mu_{A \to \varphi}(A) \times \mu_{B \to \varphi}(B) \times \mu_{C \to \varphi}(C) \Big)$$

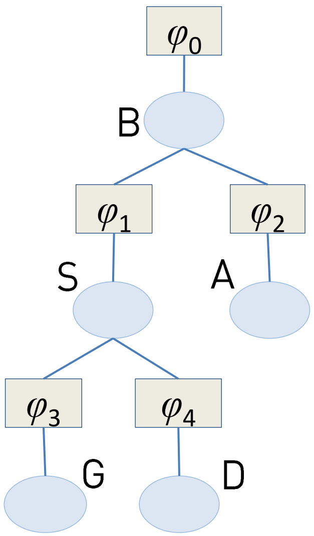
Belief Propagation: Operation

- Assume the factor graph is a tree.
- Pick an arbitrary node to be the root in the tree (i.e., a factor or a variable).
- Initialize as follows:
  if φ is a leaf node (whose parent is variable v) then $\mu_{\varphi \to \mathrm{v}}(\mathrm{v}) = \varphi(\mathrm{v})$

  if v is a leaf node (whose parent is a factor φ) then $\mu_{\mathrm{v} \to \varphi}(\mathrm{v}) = 1$
- Proceed by sending messages around. A node can send out a message if all incoming messages have arrived
- Once a node X receives all messages, we can compute the marginal:

$$P(X) = \Pi\, \mu_{\varphi \to \mathrm{v}}(\mathrm{v}) \qquad (\varphi \text{ is a neighbor of } X)$$

Belief Propagation: Compute $P(B=0 \mid G = 1, D = 0, A = 1)$



φ_0 (B)

B	val
0	0.99
1	0.01

φ_1 (S,B)

S	B	val
0	0	0.5
0	1	0.1
1	0	0.5
1	1	0.9

φ_2 (A,B)

A	B	val
0	0	0.99
0	1	0.1
1	0	0.01
1	1	0.9

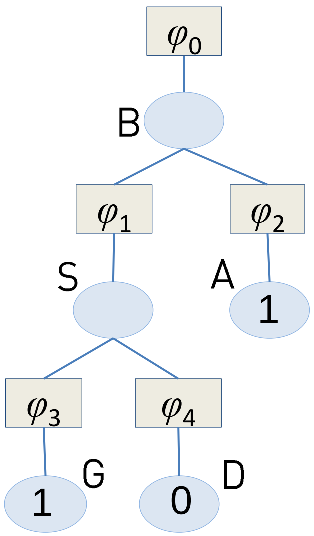
φ_3 (G,S)

G	S	val
0	0	0.7
0	1	0.4
1	0	0.3
1	1	0.6

φ_4 (D,S)

D	S	val
0	0	0.8
0	1	0.3
1	0	0.2
1	1	0.7

Belief Propagation: Compute P(B=0 | G = 1, D = 0, A = 1)



φ_0 (B)

B	val
0	0.99
1	0.01

φ_1 (S,B)

S	B	val
0	0	0.5
0	1	0.1
1	0	0.5
1	1	0.9

φ_2 (A,B)

A	B	val
0	0	0.99
0	1	0.1
1	0	0.01
1	1	0.9

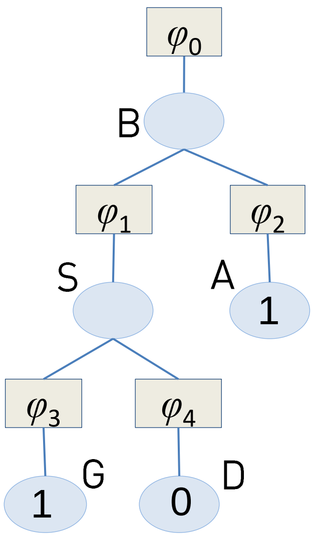
φ_3 (G,S)

G	S	val
0	0	0.7
0	1	0.4
1	0	0.3
1	1	0.6

φ_4 (D,S)

D	S	val
0	0	0.8
0	1	0.3
1	0	0.2
1	1	0.7

Belief Propagation: Compute P(B=0 | G = 1, D = 0, A = 1)



Factor-to-Variable Messages:

$$\mu_{\varphi 3 \to S}(S = 0) = \sum_{G} \varphi_3(G, S = 0) \times \mu_{G \to \varphi 3}(G) \qquad \mu_{\varphi 3 \to S}(S = 1) = \sum_{G} \varphi_3(G, S = 1) \times \mu_{G \to \varphi 3}(G)$$

$$\mu_{\varphi 4 \to S}(S = 0) = \sum_{D} \varphi_4(D, S = 0) \times \mu_{D \to \varphi 4}(D) \qquad \mu_{\varphi 4 \to S}(S = 1) = \sum_{D} \varphi_4(D, S = 1) \times \mu_{D \to \varphi 4}(D)$$

$$\mu_{\varphi 2 \to B}(B = 0) = \sum_{A} \varphi_2(A, B = 0) \times \mu_{A \to \varphi 2}(A) \qquad \mu_{\varphi 2 \to B}(B = 1) = \sum_{A} \varphi_2(A, B = 1) \times \mu_{A \to \varphi 2}(A)$$

$$\mu_{\varphi 1 \to B}(B = 0) = \sum_{S} \varphi_1(S, B = 0) \times \mu_{S \to \varphi 1}(S) \qquad \mu_{\varphi 1 \to B}(B = 1) = \sum_{S} \varphi_1(S, B = 1) \times \mu_{S \to \varphi 1}(S)$$

$$\mu_{\varphi 0 \to B}(B = 0) = \varphi_0(B = 0) \qquad \mu_{\varphi 0 \to B}(B = 1) = \varphi_0(B = 1)$$

Variable-to-Factor Messages:

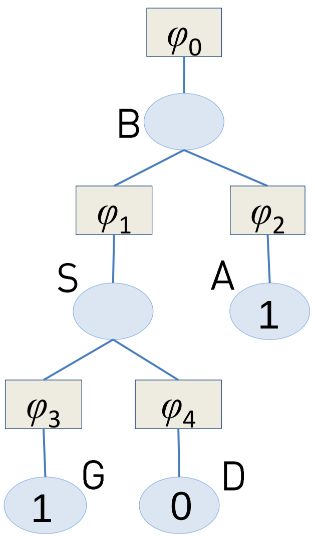
$$\mu_{G \to \varphi 3}(G = 0) = 0 \qquad \mu_{D \to \varphi 4}(D = 0) = 1 \qquad \mu_{A \to \varphi 2}(A = 0) = 0$$

$$\mu_{G \to \varphi 3}(G = 1) = 1 \qquad \mu_{D \to \varphi 4}(D = 1) = 0 \qquad \mu_{A \to \varphi 2}(A = 1) = 1$$

$$\mu_{S \to \varphi 1}(S = 0) = \mu_{\varphi 3 \to S}(S = 0) \times \mu_{\varphi 4 \to S}(S = 0)$$

$$\mu_{S \to \varphi 1}(S = 1) = \mu_{\varphi 3 \to S}(S = 1) \times \mu_{\varphi 4 \to S}(S = 1)$$

Belief Propagation: Compute P(B=0 | G = 1, D = 0, A = 1)



$$P(B = 0 \mid G=1, D=0, A=1) = \mu_{\varphi 0 \rightarrow B}(B = 0) \times \mu_{\varphi 1 \rightarrow B}(B = 0) \times \mu_{\varphi 2 \rightarrow B}(B = 0)$$

$$=$$

$$\varphi_0(B = 0) \times \left(\sum_S \varphi_1(S, B = 0) \times \mu_{S \rightarrow \varphi 1}(S) \right) \times \left(\sum_A \varphi_2(A, B = 0) \times \mu_{A \rightarrow \varphi 2}(A) \right)$$

$$=$$

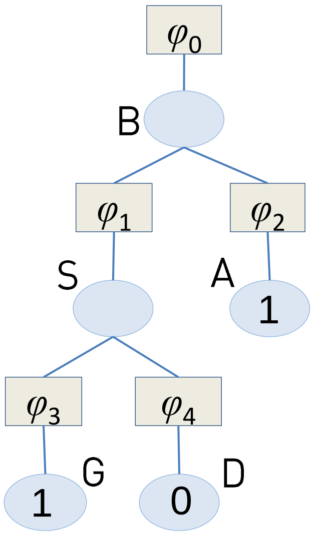
$$\varphi_0(B = 0) \times \left(\sum_S \varphi_1(S, B = 0) \times (\mu_{\varphi 3 \rightarrow S}(S) \times \mu_{\varphi 4 \rightarrow S}(S)) \right) \times \left(\sum_A \varphi_2(A, B = 0) \times (\mu_{A \rightarrow \varphi 2}(A) \right)$$

$$=$$

$$\varphi_0(B = 0) \times \left(\sum_S \varphi_1(S, B = 0) \times \left(\left(\sum_G \varphi_3(G, S) \times \mu_{G \rightarrow \varphi 3}(G)\right) \times \left(\sum_D \varphi_4(D, S) \times \mu_{D \rightarrow \varphi 4}(D)\right)\right)\right)$$

$$\times \left(\sum_A \varphi_2(A, B = 0) \times (\mu_{A \rightarrow \varphi 2}(A)) \right)$$

Belief Propagation: Compute $P(B=0 \mid G=1, D=0, A=1)$



φ_0 (B)

B	val
0	0.99
1	0.01

φ_1 (S,B)

S	B	val
0	0	0.5
0	1	0.1
1	0	0.5
1	1	0.9

φ_2 (A,B)

A	B	val
0	0	0.99
0	1	0.1
1	0	0.01
1	1	0.9

φ_3 (G,S)

G	S	val
0	0	0.7
0	1	0.4
1	0	0.3
1	1	0.6

φ_4 (D,S)

D	S	val
0	0	0.8
0	1	0.3
1	0	0.2
1	1	0.7

$$
\begin{aligned}
&P(B = 0 \mid G=1, D=0, A=1) = \\
&= 0.99 \times \\
&\quad (\varphi_1(S=0, B=0) \times \\
&\quad (\varphi_3(G=0, S=0) \times \mu_{G \to \varphi 3}(G=0) + \varphi_3(G=1, S=0) \times \mu_{G \to \varphi 3}(G=1)) \times \\
&\quad (\varphi_4(D=0, S=0) \times \mu_{D \to \varphi 4}(D=0) + \varphi_4(D=1, S=0) \times \mu_{D \to \varphi 4}(D=1))) \\
&\quad + \\
&\quad (\varphi_1(S=1, B=0) \times \\
&\quad (\varphi_3(G=0, S=1) \times \mu_{G \to \varphi 3}(G=0) + \varphi_3(G=1, S=1) \times \mu_{G \to \varphi 3}(G=1)) \times \\
&\quad (\varphi_4(D=0, S=1) \times \mu_{D \to \varphi 4}(D=0) + \varphi_4(D=1, S=1) \times \mu_{D \to \varphi 4}(D=1))) \\
&\quad \times \\
&\quad (\varphi_2(A=0, B=0) \times (\mu_{A \to \varphi 2}(A=0) + \varphi_2(A=1, B=0) \times (\mu_{A \to \varphi 2}(A=1)))
\end{aligned}
$$

$$
\begin{aligned}
&= 0.99 \times ((0.5 \times (0.7 \times 0 + 0.3 \times 1) \times (0.8 \times 1 + 0.2 \times 0) + \\
&\qquad (0.5 \times (0.4 \times 0 + 0.6 \times 1) \times (0.3 \times 1 + 0.7 \times 0)) \\
&\qquad \times (0.99 \times 0 + 0.01 \times 1) = 0.99 \times (0.5 \times 0.3 \times 0.8 + 0.5 \times 0.6 \times 0.3) \times 0.01 = 0.002079
\end{aligned}
$$

Querying the model: MAP Inference

$$P(y \mid x) = \frac{1}{Z(x)} \exp(\boldsymbol{w}^T \boldsymbol{f}(y, x))$$

Given x , we would like to predict $y = y_1 , y_2 \ldots y_n$ that maximizes $P(y \mid x)$

This requires us to make a joint prediction , together for all $y_1 , y_2 \ldots y_n$

Querying the model: MAP Inference

$$P(y \mid x) = \frac{1}{Z(x)} \exp(\boldsymbol{w}^T \boldsymbol{f}(y, x))$$

Given x , we would like to predict $y = y_1 , y_2 \ \dots\ y_n$ that maximizes $P(y \mid x)$

This requires us to make a joint prediction , together for all $y_1 , y_2 \ \dots\ y_n$

$$y^{\text{best}} = \underset{y \in \Omega_x}{\text{argmax}}\ P(y \mid x) = \underset{y \in \Omega_x}{\text{argmax}}\ \boldsymbol{w}^T \boldsymbol{f}(y, x)$$

Querying the model: MAP Inference

$$P(y \mid x) = \frac{1}{Z(x)} \exp(\boldsymbol{w}^T \boldsymbol{f}(y, x))$$

Given x , we would like to predict $y = y_1 , y_2 \; ... \; y_n$ that maximizes $P(y \mid x)$

This requires us to make a joint prediction , together for all $y_1 , y_2 \; ... \; y_n$

$$y^{\text{best}} = \underset{y \in \Omega_x}{\operatorname{argmax}} P(y \mid x) = \underset{y \in \Omega_x}{\operatorname{argmax}} \boldsymbol{w}^T \boldsymbol{f}(y, x)$$

We designed an approximate one to fit our needs , i.e., deal with many values

Querying the model: Max-marginals

$y_1^{\text{best}} = \text{argmax}\ P(y_1 | x)$ .... $y_n^{\text{best}} = \text{argmax}\ P(y_n | x)$

$$y^{\text{best}} = (y_1^{\text{best}}, \ldots, y_n^{\text{best}})$$

$\sum\prod$ belief propagation algorithms answer max-marginal queries

Querying the model: Max-marginals

y_1^{best} = argmax $P(y_1 | x$ … argmax $P(y_n | x)$

y^{best} … st$)$

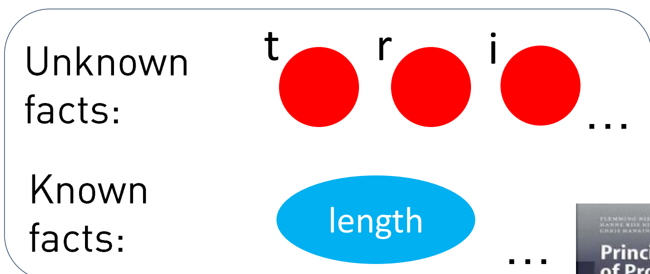
$\sum\prod$ be… pagation algorithms answer max-marginal queries

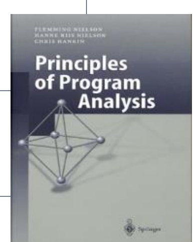
MAP inference

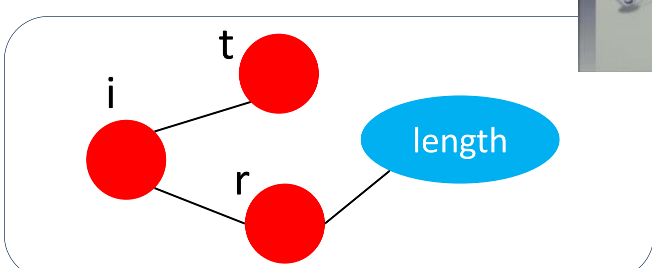
```
function chunkData(e, t)
  var n = [];
  var r = e.length;
  var i = 0;
  for (; i < r; i += t)
    if (i + t < r)
      n.push(e.substring(i, i + t));
    else
      n.push(e.substring(i, r));
  return n;
```

MAP inference

```
function chunkData(e, t)
  var n = [];
  var r = e.length;
  var i = 0;
  for (; i < r; i += t)
    if (i + t < r)
      n.push(e.substring(i, i + t));
    else
      n.push(e.substring(i, r));
  return n;
```





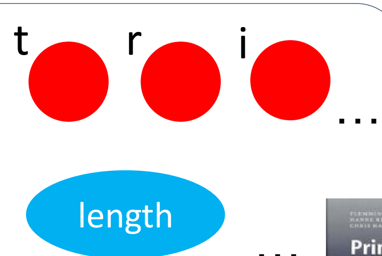


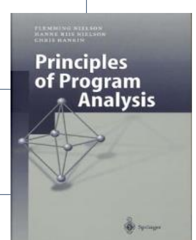
MAP inference

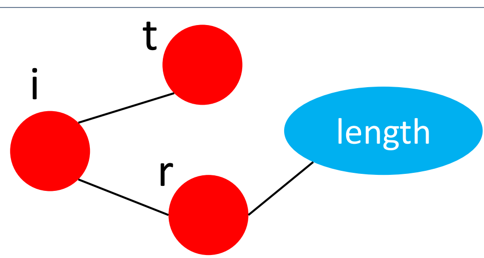
```
function chunkData(e, t)
  var n = [];
  var r = e.length;
  var i = 0;
  for (; i < r; i += t)
    if (i + t < r)
      n.push(e.substring(i, i + t));
    else
      n.push(e.substring(i, r));
  return n;
```

Unknown facts:

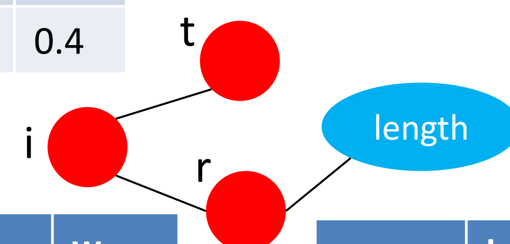
Known facts:







i	t	w
i	step	0.5
j	step	0.4

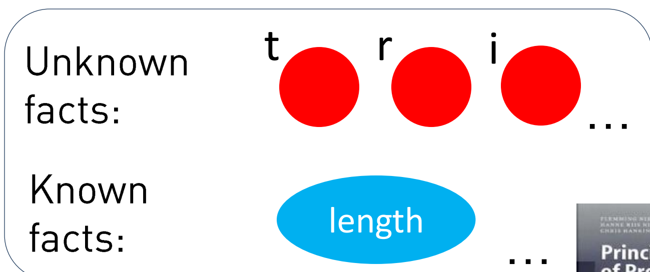


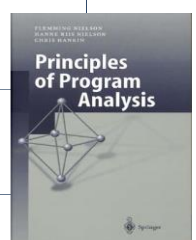
i	r	w
i	len	0.6
i	length	0.3

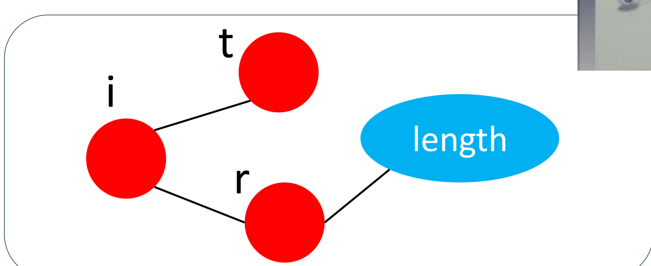
r	length	w
length	length	0.5
len	length	0.3

MAP inference

```
function chunkData(e, t)
  var n = [];
  var r = e.length;
  var i = 0;
  for (; i < r; i += t)
    if (i + t < r)
      n.push(e.substring(i, i + t));
    else
      n.push(e.substring(i, r));
  return n;
```

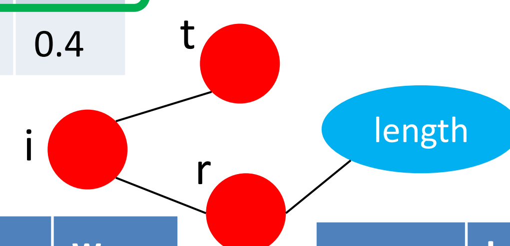






argmax $\boldsymbol{w}^T \boldsymbol{f}(i, t, r, length)$

i	t	w
i	step	0.5
j	step	0.4

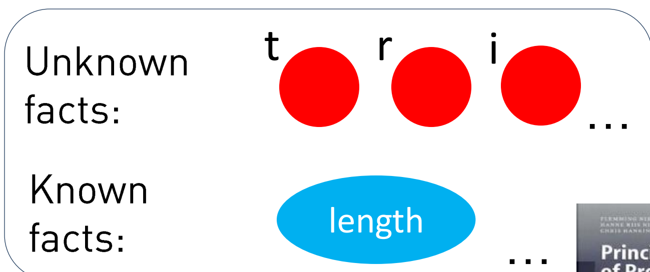


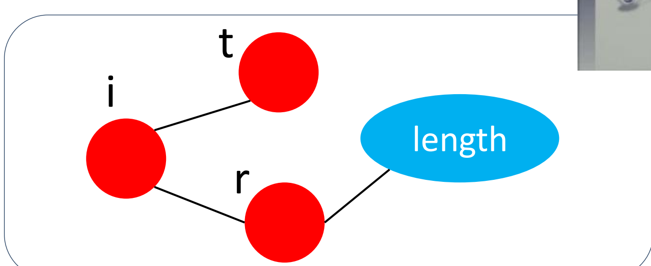
i	r	w
i	len	0.6
i	length	0.3

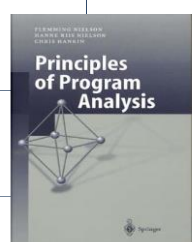
r	length	w
length	length	0.5
len	length	0.3

MAP inference

```
function chunkData(e, t)
  var n = [];
  var r = e.length;
  var i = 0;
  for (; i < r; i += t)
    if (i + t < r)
      n.push(e.substring(i, i + t));
    else
      n.push(e.substring(i, r));
  return n;
```

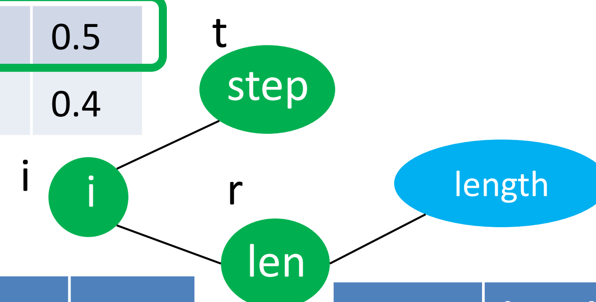






argmax $\boldsymbol{w}^T \boldsymbol{f}(i, t, r, length)$

i	t	w
i	step	0.5
j	step	0.4

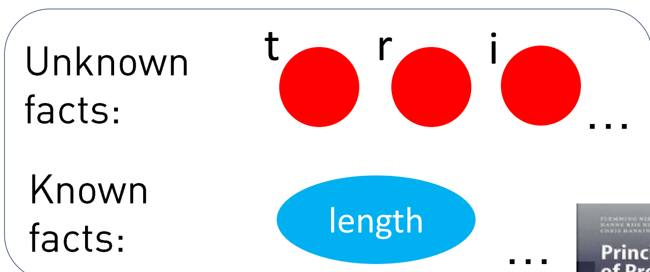


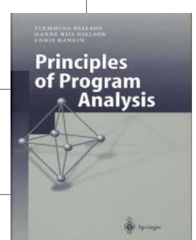
i	r	w
i	len	0.6
i	length	0.3

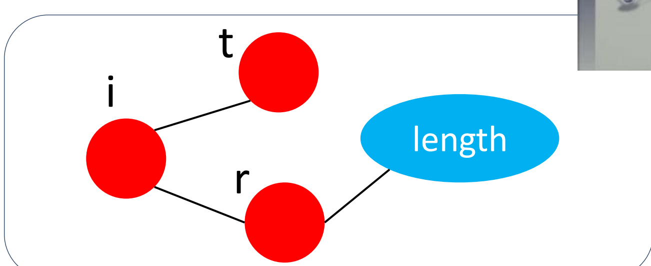
r	length	w
length	length	0.5
len	length	0.3

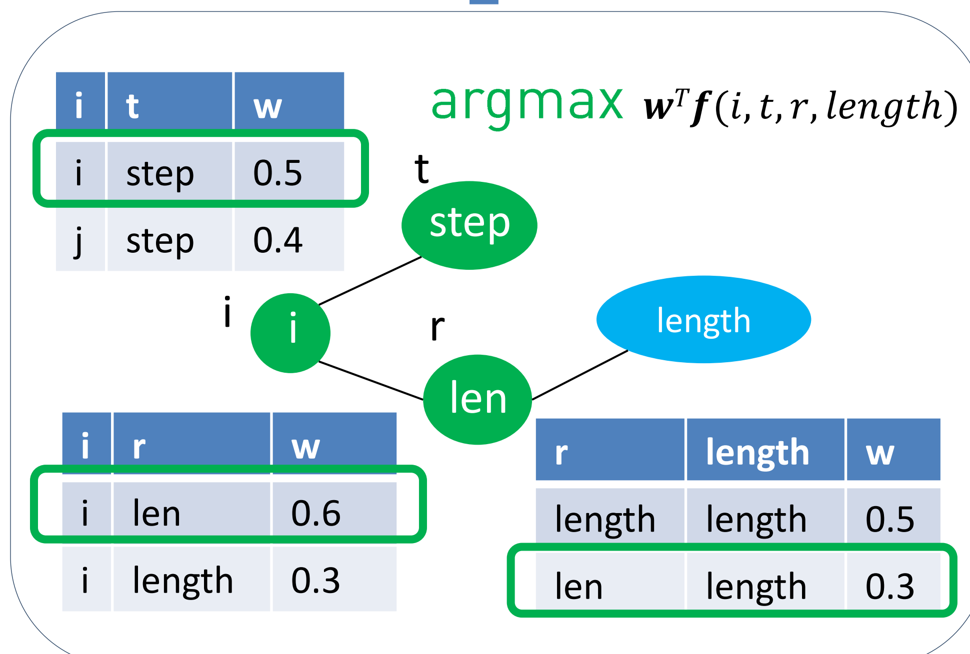
MAP inference

```
function chunkData(e, t)
   var n = [];
   var r = e.length;
   var i = 0;
   for (; i < r; i += t)
     if (i + t < r)
        n.push(e.substring(i, i + t));
    else
        n.push(e.substring(i, r));
   return n;
```









i	t	w
i	step	0.5
j	step	0.4

i	r	w
i	len	0.6
i	length	0.3

r	length	w
length	length	0.5
len	length	0.3

```
function chunkData(str, step)
   var colNames = [];
   var len = str.length;
   var i = 0;
   for (; i < len; i += step)
     if (i + step < len)
        colNames.push(str.substring(i, i + step));
    else
        colNames.push(str.substring(i, len));
   return colNames;
```

Tutorial Outline

- ~~Motivation~~
  - ~~Potential applications~~
- ~~Statistical language models~~
  - ~~N-gram and Recurrent Networks, Smoothing~~
  - ~~Application: code completion~~
- Graphical Models
  - ~~Markov Networks, Conditional Random Fields~~
  - ~~Inference in Markov Networks~~
  - Learning in Markov Networks
  - Application: predicting names and types
- Hands-on session with Nice2Predict

Structured SVM Training

(N. Ratliff, J. Bagnell, M. Zinkevich, AISTATS 2007)

$$P(y \mid x) = \frac{1}{Z(x)} \exp(\boldsymbol{w}^T \boldsymbol{f}(y, x))$$

Given a data set: D = { $\mathbf{x}^j$, $\mathbf{y}^j$ }$_{j=1..n}$ learn weights $\boldsymbol{w}^T$

Structured SVM Training

(N. Ratliff, J. Bagnell, M. Zinkevich, AISTATS 2007)

$$P(y \mid x) = \frac{1}{Z(x)} \exp(\boldsymbol{w}^T \boldsymbol{f}(y, x))$$

Given a data set: $D = \{ \mathbf{x}^j , \mathbf{y}^j \}_{j=1..n}$ learn weights $\boldsymbol{w}^T$

Optimization objective (max-margin training):

$$\forall j \; \forall \mathbf{y} \; \Sigma \, w_i f_i(\mathbf{x}^{(j)}, \mathbf{y}^{(j)}) \geq \Sigma \, w_i f_i(\mathbf{x}^{(j)}, \mathbf{y}) + \boldsymbol{\Delta}(\mathbf{y}, \mathbf{y}^{(j)})$$

Structured SVM Training

(N. Ratliff, J. Bagnell, M. Zinkevich, AISTATS 2007)

$$P(y \mid x) = \frac{1}{Z(x)} \exp(\boldsymbol{w}^T \boldsymbol{f}(y, x))$$

Given a data set: $D = \{ \mathbf{x}^j , \mathbf{y}^j \}_{j=1..n}$ learn weights $\boldsymbol{w}^T$

Optimization objective (max-margin training):

$$\forall j \ \forall \mathbf{y} \ \underbrace{\Sigma\, w_i f_i(\mathbf{x}^{(j)}, \mathbf{y}^{(j)})} \geq \underbrace{\Sigma\, w_i f_i(\mathbf{x}^{(j)}, \mathbf{y})} + \underbrace{\boldsymbol{\Delta}(\mathbf{y}, \mathbf{y}^{(j)})}$$

for all samples

Given prediction is better than **any other prediction** by a **margin**

Structured SVM Training

(N. Ratliff, J. Bagnell, M. Zinkevich, AISTATS 2007)

$$P(y \mid x) = \frac{1}{Z(x)} \exp(\boldsymbol{w}^T \boldsymbol{f}(y, x))$$

Given a data set: $D = \{ \mathbf{x}^j , \mathbf{y}^j \}_{j=1..n}$ learn weights $\boldsymbol{w}^T$

Optimization objective (max-margin training):

$$\forall j \; \forall \mathbf{y} \; \underbrace{\Sigma \, w_i f_i(\mathbf{x}^{(j)},\mathbf{y}^{(j)})} \geq \underbrace{\Sigma \, w_i f_i(\mathbf{x}^{(j)},\mathbf{y})} + \underbrace{\boldsymbol{\Delta}(\mathbf{y},\mathbf{y}^{(j)})}$$

for all samples

Given prediction is better than **any other prediction** by a **margin**

Avoids expensive computation of the partition function $Z(x)$

Tutorial Outline

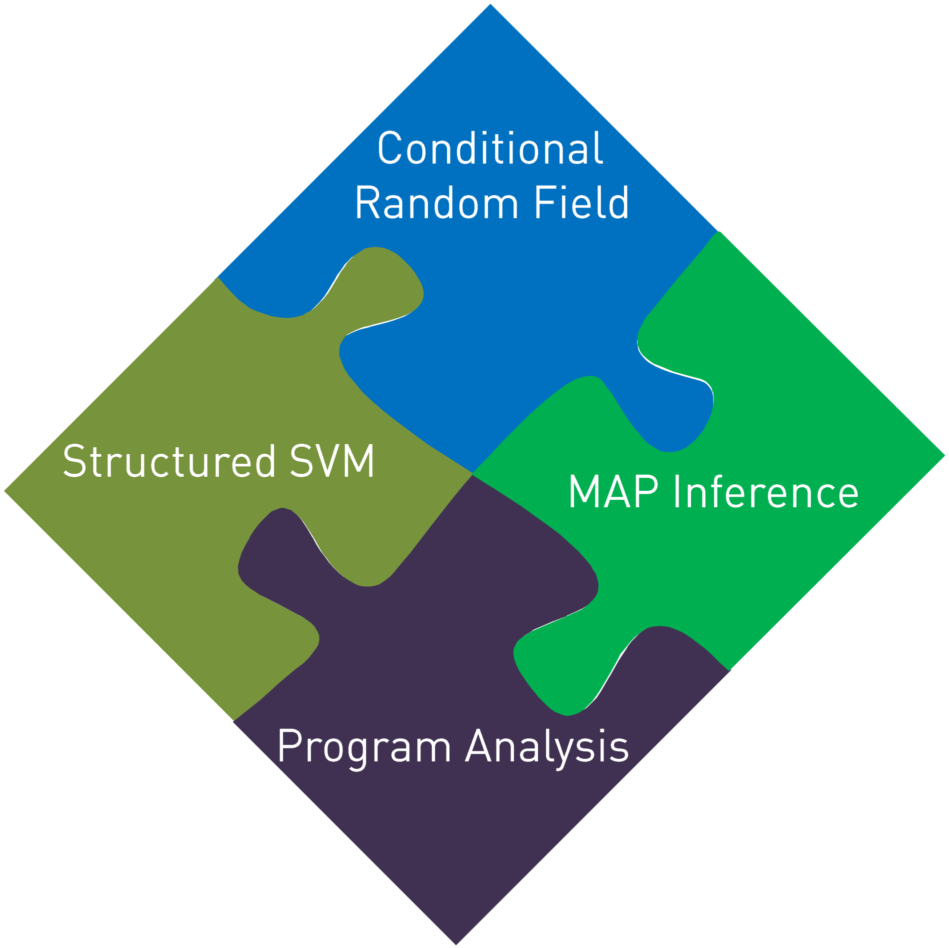
- ~~Motivation~~
  - ~~Potential applications~~
- ~~Statistical language models~~
  - ~~N-gram and Recurrent Networks, Smoothing~~
  - ~~Application: code completion~~
- Graphical Models
  - ~~Markov Networks, Conditional Random Fields~~
  - ~~Inference in Markov Networks~~
  - ~~Learning in Markov Networks~~
  - Application: predicting names and types
- Hands-on session with Nice2Predict

Tutorial Outline

- ~~Motivation~~
  - ~~Potential applications~~
- ~~Statistical language models~~
  - ~~N-gram and Recurrent Networks, Smoothing~~
  - ~~Application: code completion~~
- Graphical Models
  - ~~Markov Networks, Conditional Random Fields~~
  - ~~Inference in Markov Networks~~
  - ~~Learning in Markov Networks~~
  - Application: predicting names and types
- Hands-on session with Nice2Predict

Structured Prediction for Programs

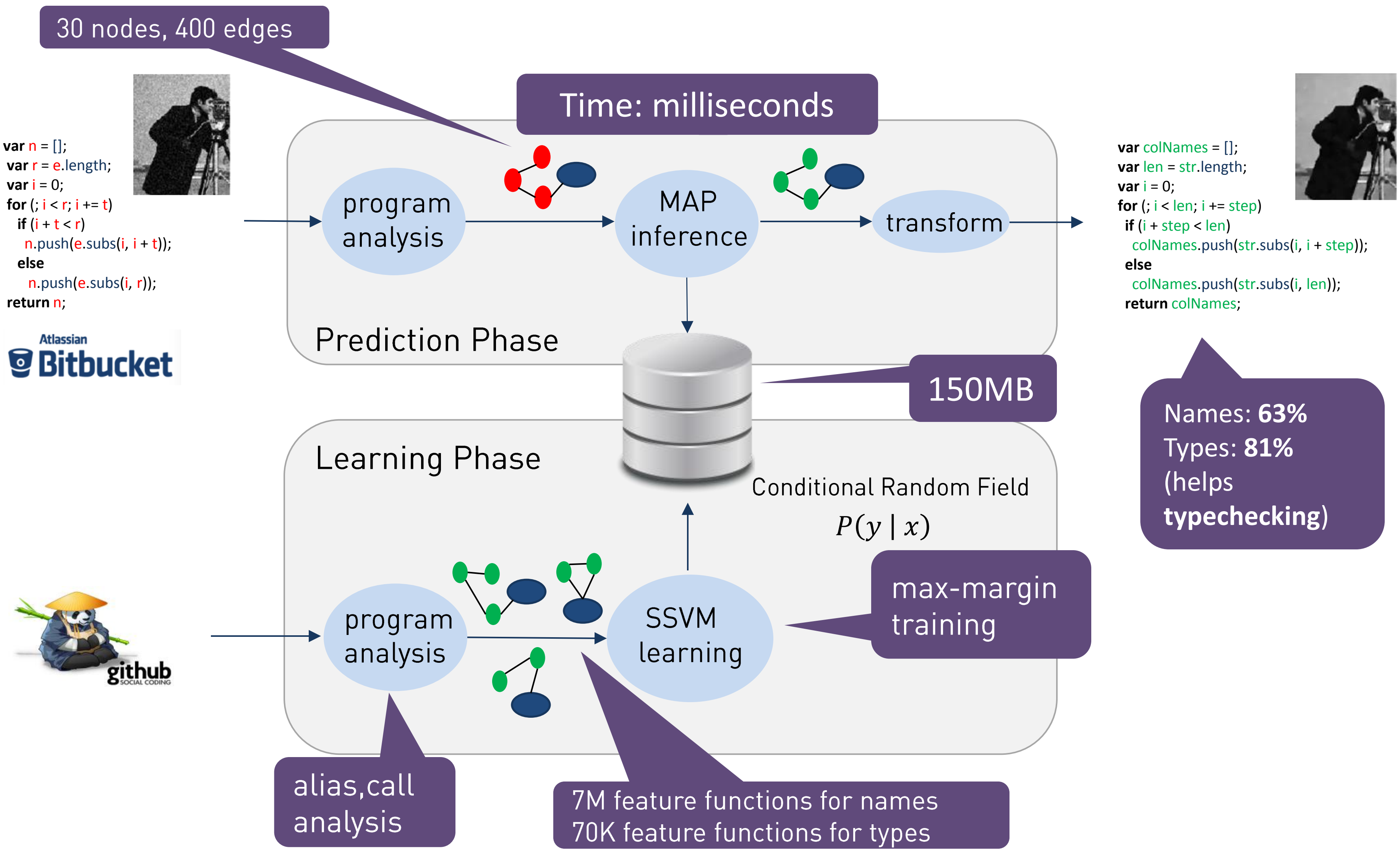
(V. Raychev, M. V., A. Krause, ACM POPL'15)



First connection between Programs and Conditional Random Fields

Structured Prediction for Programs

(V. Raychev, M. V., A. Krause, ACM POPL'15)



Machine Learning for Programming

Applications	Code completion; Deobfuscation; Program synthesis; Feedback generation; Translation
Intermediate Representation	Sequences (sentences); Trees; Translation Table; Graphical Models (CRFs); Feature Vectors
Analyze Program (PL)	typestate analysis; scope analysis; control-flow analysis; alias analysis
Train Model (ML)	Neural Networks; N-gram language model; SVM; Structured SVM
Query Model (ML)	argmax P(y | x) y ∈ Ω; Greedy MAP inference

More information and tutorials at:

http://www.nice2predict.org/
http://www.srl.inf.ethz.ch/

ETH Zürich