Joint Inference for the NLP Pipeline
Probabilistic Programming and the FACTORIE System

Andrew McCallum and **David Smith**
Karl Schultz, Khashayar Rohanimanesh,
Sameer Singh, Michael Wick
Outline

- The Subtle pipeline
- Obstacles to joint inference
- FACTORIE: Declarative semantics, procedural definitions
- Efficient training with SampleRank and reinforcement learning
The Subtle Pipeline
The Subtle Pipeline

- Parsing
- Semantics
- Pragmatics
- World Model
- PAR + LTL

Cascading Errors

- Natural Language Processing
- Linguistics
- Robotics
- Graphics/Human Simulation
- Machine Learning

Parse tree, indices, semantic tags

Underspecified predicate logic
The Subtle Pipeline

Joint Inference

World Model

PAR + LTL

Natural Language Processing

Linguistics

Robotics

Graphics/ Human Simulation

Parsing

Semantics

Pragmatics

Parse tree, indices, semantic tags

Underspecified predicate logic

Friday, October 16, 2009
The Subtle Pipeline

- Parsing
- Semantics
- Pragmatics
- World Model
- PAR + LTL

Joint Inference

- Parse tree, indices, semantic tags
- Underspecified predicate logic

Natural Language Processing
Linguistics
Robotics
Graphics/Human Simulation

Generally Intractable
Obstacles to Joint Inference
Conditional Random Fields

Undirected graphical model, trained to maximize conditional probability of output (sequence) given input (sequence)

\[ p(\vec{s} | \vec{o}) = \frac{1}{Z_{\vec{o}}} \prod_{t=1}^{\mid \vec{o} \mid} \phi(s_t, s_{t-1}) \phi(o_t, s_t) \]

transitions observations

\([\text{Lafferty, McCallum, Pereira 2001}]\)
Conditional Random Fields

Undirected graphical model, trained to maximize conditional probability of output (sequence) given input (sequence)

\[
p(\bar{s} | \bar{o}) = \frac{1}{Z_{\bar{o}}} \prod_{t=1}^{|ar{o}|} \phi(s_t, s_{t-1}) \phi(o_t, s_t) = \exp \left( \sum_{k} \lambda_k f_k(o_t, s_t) \right)
\]
Conditional Random Fields

Undirected graphical model, trained to maximize conditional probability of output (sequence) given input (sequence)

Finite state model

Graphical model

\[
p(\vec{s} | \vec{o}) = \frac{1}{Z_{\vec{o}}} \prod_{t=1}^{\lvert \vec{o} \rvert} \phi(s_t, s_{t-1}) \phi(o_t, s_t) = \exp \left( \sum_k \lambda_k f_k(o_t, s_t) \right)
\]
Conditional Random Fields

Undirected graphical model, trained to maximize conditional probability of output (sequence) given input (sequence)

\[ p(\vec{s} | \vec{o}) = \frac{1}{Z_{\vec{o}}} \prod_{t=1}^{\vec{o}} \phi(s_t, s_{t-1}) \phi(o_t, s_t) = \exp \left( \sum_k \lambda_k f_k(o_t, s_t) \right) \]

- Named-entity tag
- Noun-phrase boundaries
- Part-of-speech
- English words

\( o_1 \quad o_2 \quad o_3 \quad o_4 \quad o_5 \)
Conditional Random Fields

Undirected graphical model, trained to maximize conditional probability of output (sequence) given input (sequence)

\[ p(\mathbf{s}|\mathbf{o}) = \frac{1}{Z_{\mathbf{o}}} \prod_{t=1}^{\mid\mathbf{o}\mid} \phi(s_t, s_{t-1}) \phi(o_t, s_t) \]

\[ = \exp \left( \sum_k \lambda_k f_k(o_t, s_t) \right) \]

\( O(T^{|V|}) \) variable assignments

- Named-entity tag
- Noun-phrase boundaries
- Part-of-speech
- English words
Message Passing

adapted from MacKay (2003) textbook
Message Passing

*Count the soldiers*

adapted from MacKay (2003) textbook
Message Passing

Count the soldiers

adapted from MacKay (2003) textbook
Message Passing

*Count the soldiers*

adapted from MacKay (2003) textbook
Message Passing

Count the soldiers

there's 1 of me

5 behind you
4 behind you
3 behind you
2 behind you
1 behind you

adapted from MacKay (2003) textbook
Message Passing

Count the soldiers

adapted from MacKay (2003) textbook
Message Passing

Count the soldiers

adapted from MacKay (2003) textbook
Message Passing

1 of me

2 before you

only see my incoming messages

3 behind you

adapted from MacKay (2003) textbook
Message Passing

only see my incoming messages

2 before you

3 behind you

there's 1 of me

Belief: Must be $2 + 1 + 3 = 6$ of us

adapted from MacKay (2003) textbook
Message Passing

Count the soldiers

2 before you

Belief:
Must be
2 + 1 + 3 = 6
of us

there's 1 of me

only see my incoming messages

3 behind you

adapted from MacKay (2003) textbook
Message Passing

Belief:
Must be
2 + 1 + 3 = 6
of us

there's 1 of me

1 before you

only see my incoming messages

4 behind you

adapted from MacKay (2003) textbook
Message Passing

Count the soldiers:

1 before you

only see my incoming messages

there's 1 of me

Belief:
Must be 1 + 1 + 4 = 6 of us

adapted from MacKay (2003) textbook
Message Passing

Each soldier receives reports from all branches of tree

adapted from MacKay (2003) textbook
Message Passing

Each soldier receives reports from all branches of the tree.

Belief: Must be 14 of us.

Wouldn’t work correctly with a “loopy” (cyclic) graph.

adapted from MacKay (2003) textbook
Sampling
Sampling
Coreference Resolution

AKA "record linkage", "database record deduplication", "citation matching", "object correspondence", "identity uncertainty"

Input

News article, with named-entity "mentions" tagged

Output

Number of entities, $N = 3$

#1

Secretary of State Colin Powell

he

Mr. Powell

Powell

#2

Condoleezza Rice

she

Rice

#3

President Bush

he

Bush

he
Coreference Resolution

AKA "record linkage", "database record deduplication",
"citation matching", "object correspondence", "identity uncertainty"

Player 2: i have 4H
Player 1: I want it!
Player 1: where is it?
Player 2: should i leave it for you somewhere?
Player 1: sure
Player 1: where are you?
Player 2: okay, where are you?
Player 1: I'm near the top
Player 2: I'm left side.
Player 1: next to the gap near the middle
Player 2: I'll leave the card in the upper left corner.
Player 1: awesome
Coreference Resolution

AKA "record linkage", "database record deduplication", "citation matching", "object correspondence", "identity uncertainty"

Player 2: i have 4H
Player 1: I want it!
Player 1: where is it?
Player 2: should i leave it for you somewhere?
Player 1: sure
Player 1: where are you?
Player 2: okay, where are you?
Player 1: I'm near the top
Player 2: i'm left side.
Player 1: next to the gap near the middle
Player 2: i'll leave the card in the upper left corner.
Player 1: awesome

Combined segmentation and clustering
Pairwise Affinity is not Enough
Pairwise Affinity is not Enough
Pairwise Comparisons Not Enough

Examples:

• ∀ mentions are pronouns?
• Entities have multiple attributes \((\text{name, email, institution, location})\); need to measure “compatibility” among them.
• Having 2 “given names” is common, but not 4. —e.g. Howard M. Dean / Martin, Dean / Howard Martin
• Need to measure size of the clusters of mentions.
• ∃ a pair of lastname strings that differ > 5?

• We need to ask ∃, ∀ questions about a set of mentions
• We want first-order logic!
Pairwise Affinity is not Enough
Ask arbitrary questions about all entities in a partition with first-order logic...
Partition Affinity CRF

- she
- Amy Hall
- she
- she
Partition Affinity CRF
Partition Affinity CRF
Partition Affinity CRF

she

she

she

Amy Hall
Factors quartic in # of tokens

Factors quadratic in # of mentions

Unrolled graph instantiates all alignments
This space complexity is common in probabilistic first-order logic models
How can we perform inference and learning in models that cannot be grounded?
Don’t represent all alternatives...
Don’t represent all alternatives... just one at a time
Don’t represent all alternatives... just one at a time
Don’t represent all alternatives... just one at a time
FACTORIE: Declarative Semantics, Procedural Definition
Declarative Model Specification
Declarative Model Specification

• One of biggest advances in AI & ML
Declarative Model Specification

- One of biggest advances in AI & ML
- Gone too far? Much domain knowledge is also procedural.
Declarative Model Specification

• One of biggest advances in AI & ML

• Gone too far? Much domain knowledge is also procedural.

• Logic + Probability $\rightarrow$ Imperative + Probability
  – Rising interest: Church, Csoft,...

• Our approach
  – Preserve the *declarative* statistical semantics of factor graphs
  – “Imperatively-Defined Factor Graphs” (IDFs)
Our Design Goals
Our Design Goals

• Represent factor graphs
  – emphasis on discriminative undirected models
Our Design Goals

• Represent factor graphs
  – emphasis on discriminative undirected models

• Scalability
  – input data, output configuration, factors, tree-width
  – observed data that cannot fit in memory
  – exponential number of factors
Our Design Goals

• Represent factor graphs
  – emphasis on discriminative undirected models

• Scalability
  – input data, output configuration, factors, tree-width
  – observed data that cannot fit in memory
  – exponential number of factors

• Efficient discriminative parameter estimation
  – sensitive to the expense of inference
Our Design Goals

• Represent factor graphs
  – emphasis on discriminative undirected models

• Scalability
  – input data, output configuration, factors, tree-width
  – observed data that cannot fit in memory
  – exponential number of factors

• Efficient discriminative parameter estimation
  – sensitive to the expense of inference

• Integrate declarative & procedural knowledge
  – natural, easy-to-use
  – upcoming: 3 examples of injecting imperativ-ism into factor graphs
• Factor Graphs, Imperative, Extensible

• Implemented as a library in Scala [Martin Odersky]
  - object oriented & functional
  - type inference
  - lazy evaluation
  - everything an object (int, float,...)
  - nice syntax for creating “domain-specific languages”
  - runs in JVM (complete interoperation with Java)
  - “Haskell++ in a Java style”

• Library, not new “little language”
  - all familiar Java constructs & libraries available to you
  - integrate data pre-processing & eval. w/ model spec
  - Scala makes syntax not too bad.
  - But not as compact as a dedicated language (BLOG, MLN)
Example: Linear-Chain CRF for Segmentation

class Label(isBeg: boolean) extends Bool(isBeg)

class Token(word: String) extends EnumVariable(word)
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends Bool(isBeg) with VarSeq {
  val token : Token
}
class Token(word:String) extends EnumVariable(word) with VarSeq {
  val label : Label
}
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends Bool(isBeg) with VarSeq {
    val token : Token
}
class Token(word:String) extends EnumVariable(word) with VarSeq {
    val label : Label
}

Avoid representing relations by indices.
Do it directly with members, pointers... arbitrary data structure.

```
Bill    loves        skiing Tom     loves    snowshoeing
T      F            F                 T      F           F

Labels

Words
Bill   loves   skiing   Tom   loves   snowshoeing
```
Example: Linear-Chain CRF for Segmentation

class Label(isBeg: boolean) extends Bool(isBeg) with VarSeq {
    val token : Token
}
class Token(word: String) extends EnumVariable(word) with VarSeq {
    val label : Label
    def longerThanSix = word.length > 6
}
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends Bool(isBeg) with VarSeq {
    val token : Token
}
class Token(word:String) extends EnumVariable(word) with VarSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

// Factor templates
object StateTemplate extends Template1[Label]
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends Bool(isBeg) with VarSeq {
    val token : Token
}
class Token(word: String) extends EnumVariable(word) with VarSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

// Factor templates
object StateTemplate extends Template1[Label]
object StateTokenTemplate extends Template2[Label, Token]
Example: Linear-Chain CRF for Segmentation

class Label(isBeg: boolean) extends Bool(isBeg) with VarSeq {
    val token: Token
}
class Token(word: String) extends EnumVariable(word) with VarSeq {
    val label: Label
    def longerThanSix = word.length > 6
}

// Factor templates
object StateTemplate extends Template1[Label]
object StateTokenTemplate extends Template2[Label, Token]
object StateTransitionTemplate extends Template2[Label, Label]
Imperativ-ism #1: Jump Function

• Proposal “jump function”
  – Make changes to world state

• Sometimes simple, sometimes not
  – Sample Gaussian with mean at old value
  – Sample cluster to split, run stochastic greedy agglomerative clustering

• Gibbs sampling, one variable at a time
  – poor mixing

• Rich jump function
  – Natural place to embed domain knowledge about what variables should change in concert.
Imperativ-ism #1: Jump Function

- Proposal “jump function”
  - Make changes to world state

- Sometimes simple, sometimes not
  - Sample Gaussian with mean at old value
  - Sample cluster to split, run stochastic greedy agglomerative clustering

- Gibbs sampling, one variable at a time
  - Poor mixing

- Rich jump function
  - Natural place to embed domain knowledge about what variables should change in concert.
  - Avoid some expensive deterministic factors with property-preserving jump functions (e.g. coref transitivity, dependency parsing projectivity)
Key Operation: Scoring a Proposal

- Acceptance probability ~ ratio of model scores. Scores of factors that didn’t change cancel.
- To efficiently score:
  - Proposal method runs.
Key Operation: Scoring a Proposal

• Acceptance probability ~ ratio of model scores. Scores of factors that didn’t change cancel.

• To efficiently score:
  – Proposal method runs.
  – Automatically build a list of variables that changed.
Key Operation: Scoring a Proposal

- Acceptance probability ~ ratio of model scores. Scores of factors that didn’t change cancel.
- To efficiently score:
  - Proposal method runs.
  - Automatically build a list of variables that changed.
  - Find factors that touch changed variables
Key Operation: Scoring a Proposal

- Acceptance probability ~ ratio of model scores. Scores of factors that didn’t change cancel.
- To efficiently score:
  - Proposal method runs.
  - Automatically build a list of variables that changed.
  - Find factors that touch changed variables
  - Find other (unchanged) variables needed to calculate those factors’ scores
Key Operation: Scoring a Proposal

• Acceptance probability ~ ratio of model scores. Scores of factors that didn’t change cancel.

• To efficiently score:
  – Proposal method runs.
  – Automatically build a list of variables that changed.
  – Find factors that touch changed variables.
  – Find other (unchanged) variables needed to calculate those factors’ scores.
Key Operation: Scoring a Proposal

• Acceptance probability ~ ratio of model scores. Scores of factors that didn’t change cancel.

• To efficiently score:
  – Proposal method runs.
  – Automatically build a list of variables that changed.
  – Find factors that touch changed variables
  – Find other (unchanged) variables needed to calculate those factors’ scores

• How to find factors from variables & vice versa?
  – In BLOG, rich, highly-indexed data structure stores mapping variables ←→ factors
  – But complex to maintain as structure changes
Imperativ-ism #2: Model Structure
Imperativ-ism #2: Model Structure

• Maintain no map structure between factors and variables
Imperativ-ism #2: Model Structure

- Maintain no map structure between factors and variables

- Finding factors is easy. Usually # templates < 50.
Imperativ-ism #2: Model Structure

• Maintain no map structure between factors and variables

• Finding factors is easy. Usually # templates < 50.

• Primitive operation:
  Given factor template and one changed variable, find other variables
Imperativ-ism #2: Model Structure

• Maintain no map structure between factors and variables

• Finding factors is easy. Usually # templates < 50.

• Primitive operation:
  Given factor template and one changed variable, find other variables

• In factor Template object, define imperative methods that do this.
  • unroll1(v1) returns (v1,v2,v3)
  • unroll2(v2) returns (v1,v2,v3)
  • unroll3(v3) returns (v1,v2,v3)
  – i.e., use Turing-complete language to determine structure on the fly.
  – If you want to use a data structure instead, access it in the method.
  – If you want a higher-level language for specifying structure, write it terms of this primitive.

• Other nice attribute
  – Easy to do value-conditioned structure. Case Factor Diagrams, etc.
  – Not only avoid unrolling, don’t even allocate all factors for current config.
Example: Linear-Chain CRF for Segmentation

class Label(isBeg: boolean) extends Bool(isBeg) with VarSeq {
    val token : Token
}
class Token(word: String) extends EnumVariable(word) with VarSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

// Factor templates
object StateTemplate extends Template1[Label]
object StateTokenTemplate extends Template2[Label, Token]
object StateTransitionTemplate extends Template2[Label, Label]
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends Bool(isBeg) with VarSeq {
    val token : Token
}
class Token(word:String) extends EnumVariable(word) with VarSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

// Factor templates
object StateTemplate extends Template1[Label]
object StateTokenTemplate extends Template2[Label,Token]

object StateTransitionTemplate extends Template2[Label,Label]
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends Bool(isBeg) with VarSeq {
  val token : Token
}
class Token(word:String) extends EnumVariable(word) with VarSeq {
  val label : Label
  def longerThanSix = word.length > 6
}

// Factor templates
object StateTemplate extends Template1[Label]
object StateTokenTemplate extends Template2[Label,Token] {
  def unroll1(label:Label) = Factor(label, label.token)
}

object StateTransitionTemplate extends Template2[Label,Label]

Labels

Words
class Label(isBeg: boolean) extends Bool(isBeg) with VarSeq {
    val token : Token
}
class Token(word: String) extends EnumVariable(word) with VarSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

// Factor templates
object StateTemplate extends Template1[Label]
object StateTokenTemplate extends Template2[Label, Token] {
    def unroll1(label: Label) = Factor(label, label.token)
    def unroll2(token: Token) = new Error // Tokens shouldn’t change
}
object StateTransitionTemplate extends Template2[Label, Label]
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends Bool(isBeg) with VarSeq {
  val token : Token
}
class Token(word:String) extends EnumVariable(word) with VarSeq {
  val label : Label
  def longerThanSix = word.length > 6
}

// Factor templates
object StateTemplate extends Template1[Label]
object StateTokenTemplate extends Template2[Label,Token] {
  def unroll1(label:Label) = Factor(label, label.token)
  def unroll2(token:Token) = new Error // Tokens shouldn’t change
}
object StateTransitionTemplate extends Template2[Label,Label] {
  def unroll1(label:Label) = Factor(label, label.next)
}
class Label(isBeg:boolean) extends Bool(isBeg) with VarSeq {
    val token : Token
}
class Token(word:String) extends EnumVariable(word) with VarSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

// Factor templates
object StateTemplate extends Template1[Label]
object StateTokenTemplate extends Template2[Label,Token] {
    def unroll1(label:Label) = Factor(label, label.token)
    def unroll2(token:Token) = new Error // Tokens shouldn’t change
}
object StateTransitionTemplate extends Template2[Label,Label] {
    def unroll1(label:Label) = Factor(label, label.next)
    def unroll2(label:Label) = Factor(label.prev, label)
}
Imperativ-ism #3: Neighbor-Sufficient Map
Imperativ-ism #3: Neighbor-Sufficient Map

• “Neighbor Variables” of a factor
  – Variables touching the factor
• “Sufficient Statistics” of a factor
  – Vector, dot product with weights of log-linear factor → factor’s score
Imperativ-ism #3: Neighbor-Sufficient Map

• “Neighbor Variables” of a factor
  – Variables touching the factor

• “Sufficient Statistics” of a factor
  – Vector, dot product with weights of log-linear factor → factor’s score

• Usually confounded. Separate them.

• Skip-chain NER. Instead of 5x5 parameters, just 2. 
  \((\text{label1, label2}) \rightarrow \text{label1 == label2}\)
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends Bool(isBeg) with VarSeq {
   val token : Token
}
class Token(word:String) extends EnumVariable(word) with VarSeq {
   val label : Label
   def longerThanSix = word.length > 6
}

// Factor templates
object StateTemplate extends Template1[Label]
object StateTokenTemplate extends Template2[Label,Token] {
   def unroll1(label:Label) = Factor(label, label.token)
   def unroll2(token:Token) = new Error // Tokens shouldn’t change
}
object StateTransitionTemplate extends Template2[Label,Label] {
   def unroll1(label:Label) = Factor(label, label.next)
   def unroll2(label:Label) = Factor(label.prev, label)
}
class Label(isBeg:boolean) extends Bool(isBeg) with VarSeq {
    val token : Token
}
class Token(word:String) extends EnumVariable(word) with VarSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

// Factor templates
object StateTemplate extends Template1[Label]
object StateTokenTemplate extends Template2[Label,Token] {
    def unroll1(label:Label) = Factor(label, label.token)
    def unroll2(token:Token) = new Error // Tokens shouldn’t change
}
object StateTransitionTemplate extends Template2[Label,Label] {
    def unroll1(label:Label) = Factor(label, label.next)
    def unroll2(label:Label) = Factor(label.prev, label)
}
object SkipTemplate extends Template1[Bool] with Neighbors[Label,Label]
Example: Skip-Chain CRF for Segmentation

class Label(isBeg:boolean) extends Bool(isBeg) with VarSeq {
    val token : Token
}
class Token(word:String) extends EnumVariable(word) with VarSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

// Factor templates
object StateTemplate extends Template1[Label]  
object StateTokenTemplate extends Template2[Label,Token] {
    def unroll1(label:Label) = Factor(label, label.token)
    def unroll2(token:Token) = new Error // Tokens shouldn’t change
}
object StateTransitionTemplate extends Template2[Label,Label] {
    def unroll1(label:Label) = Factor(label, label.next)
    def unroll2(label:Label) = Factor(label.prev, label)
}
object SkipTemplate extends Template1[Bool] with Neighbors[Label,Label]{
    def unroll1(label:Label) = for (other <- label.seq)
        if (label.token == other.token)) yield Factor (label,other)
}
class Label(isBeg:boolean) extends Bool(isBeg) with VarSeq {
    val token : Token
}
class Token(word:String) extends EnumVariable(word) with VarSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

// Factor templates
object StateTemplate extends Template1[Label]
object StateTokenTemplate extends Template2[Label,Token] {
    def unroll1(label:Label) = Factor(label, label.token)
    def unroll2(token:Token) = new Error // Tokens shouldn’t change
}
object StateTransitionTemplate extends Template2[Label,Label] {
    def unroll1(label:Label) = Factor(label, label.next)
    def unroll2(label:Label) = Factor(label.prev, label)
}
object SkipTemplate extends Template1[Bool] with Neighbors[Label,Label]{
    def unroll1(label:Label) = for (other <- label.seq;
        if (label.token == other.token)) yield Factor (label,other)
    def sufficient(label1:Label,label2:Label) = Suff(label1 == label2)
}
Extensibility

• Many variables types provided:
  – boolean, int, float, String, categorical, ...

• Create new ones!
  – set-valued variable
  – finite-state machine as a variable [JHU]

• Create new factor types
  – Poisson, Dirichlet, ...
Experimental Results

• Joint Segmentation & Coreference of research paper citations.
  – 1295 mentions, 134 entities, 36487 tokens

• Compare with MLNs (Alchemy)
  – Same observable features

• Factorie results:
  – ~25% reduction in error (segmentation & coref)
  – 3-20x faster
Experimental Results

Pairwise F1, Cluster Recall

<table>
<thead>
<tr>
<th>Method</th>
<th>Prec/Recall</th>
<th>F1</th>
<th>Cluster Rec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fellegi-Sunter</td>
<td>78.0/97.7</td>
<td>86.7</td>
<td>62.7</td>
</tr>
<tr>
<td>Joint MLN</td>
<td>94.3/97.0</td>
<td>95.6</td>
<td>78.1</td>
</tr>
<tr>
<td>Isolated IDF</td>
<td>97.09/95.42</td>
<td>96.22</td>
<td>86.01</td>
</tr>
<tr>
<td>Joint IDF</td>
<td>95.34/98.25</td>
<td>96.71</td>
<td>94.62</td>
</tr>
</tbody>
</table>

Tokenwise F1

<table>
<thead>
<tr>
<th>Method</th>
<th>Author</th>
<th>Title</th>
<th>Venue</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolated MLN</td>
<td>99.3</td>
<td>97.3</td>
<td>98.2</td>
<td>98.2</td>
</tr>
<tr>
<td>Joint MLN</td>
<td>99.5</td>
<td>97.6</td>
<td>98.3</td>
<td>98.4</td>
</tr>
<tr>
<td>Isolated IDF</td>
<td>99.35</td>
<td>97.63</td>
<td>98.58</td>
<td>98.51</td>
</tr>
<tr>
<td>Joint IDF</td>
<td>99.42</td>
<td>97.99</td>
<td>98.78</td>
<td>98.72</td>
</tr>
</tbody>
</table>
Experimental Results

Pairwise F1, Cluster Recall

<table>
<thead>
<tr>
<th>Method</th>
<th>Prec/Recall</th>
<th>F1</th>
<th>Cluster Rec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fellegi-Sunter</td>
<td>78.0/97.7</td>
<td>86.7</td>
<td>62.7</td>
</tr>
<tr>
<td>Joint MLN</td>
<td>94.3/97.0</td>
<td>95.6</td>
<td>78.1</td>
</tr>
<tr>
<td>Isolated IDF</td>
<td>97.09/95.42</td>
<td>96.22</td>
<td>86.01</td>
</tr>
<tr>
<td>Joint IDF</td>
<td>95.34/98.25</td>
<td>96.71</td>
<td>94.62</td>
</tr>
</tbody>
</table>

Tokenwise F1

<table>
<thead>
<tr>
<th>Method</th>
<th>Author</th>
<th>Title</th>
<th>Venue</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolated MLN</td>
<td>99.3</td>
<td>97.3</td>
<td>98.2</td>
<td>98.2</td>
</tr>
<tr>
<td>Joint MLN</td>
<td>99.5</td>
<td>97.6</td>
<td>98.3</td>
<td>98.4</td>
</tr>
<tr>
<td>Isolated IDF</td>
<td>99.35</td>
<td>97.63</td>
<td>98.58</td>
<td>98.51</td>
</tr>
<tr>
<td>Joint IDF</td>
<td>99.42</td>
<td>97.99</td>
<td>98.78</td>
<td>98.72</td>
</tr>
</tbody>
</table>

So Singhm Ko Schultzm Ao McCallum hUMassi Bindirectional Joint Inference

ECML PKDD 2009

50-90 min.
## Experimental Results

### Pairwise F1, Cluster Recall

<table>
<thead>
<tr>
<th>Method</th>
<th>Prec/Recall</th>
<th>F1</th>
<th>Cluster Rec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fellegi-Sunter</td>
<td>78.0/97.7</td>
<td>86.7</td>
<td>62.7</td>
</tr>
<tr>
<td>Joint MLN</td>
<td>94.3/97.0</td>
<td>95.6</td>
<td>78.1</td>
</tr>
<tr>
<td>Isolated IDF</td>
<td>97.09/95.42</td>
<td>96.22</td>
<td>86.01</td>
</tr>
<tr>
<td>Joint IDF</td>
<td>95.34/98.25</td>
<td>96.71</td>
<td>94.62</td>
</tr>
</tbody>
</table>

### Tokenwise F1

<table>
<thead>
<tr>
<th>Method</th>
<th>Author</th>
<th>Title</th>
<th>Venue</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolated MLN</td>
<td>99.3</td>
<td>97.3</td>
<td>98.2</td>
<td>98.2</td>
</tr>
<tr>
<td>Joint MLN</td>
<td>99.5</td>
<td>97.6</td>
<td>98.3</td>
<td>98.4</td>
</tr>
<tr>
<td>Isolated IDF</td>
<td>99.35</td>
<td>97.63</td>
<td>98.58</td>
<td>98.51</td>
</tr>
<tr>
<td>Joint IDF</td>
<td>99.42</td>
<td>97.99</td>
<td>98.78</td>
<td>98.72</td>
</tr>
</tbody>
</table>
Efficient Training: SampleRank and Reinforcement Learning
Parameter Estimation in Large State Spaces
Most methods require calculating gradient of log-likelihood, $P(y_1, y_2, y_3, ... \mid x_1, x_2, x_3, ...)$...
Parameter Estimation in Large State Spaces

- Most methods require calculating gradient of log-likelihood, $P(y_1, y_2, y_3, \ldots \mid x_1, x_2, x_3, \ldots)\ldots$
- ...which in turn requires “expectations of marginals,” $P(y_1 \mid x_1, x_2, x_3, \ldots)$
Parameter Estimation in Large State Spaces

- Most methods require calculating gradient of log-likelihood, \( P(y_1, y_2, y_3, \ldots \mid x_1, x_2, x_3, \ldots) \)...

- ...which in turn requires “expectations of marginals,” \( P(y_1 \mid x_1, x_2, x_3, \ldots) \)

- But, getting marginal distributions by sampling can be inefficient due to large sample space.
Parameter Estimation in Large State Spaces

- Most methods require calculating gradient of log-likelihood, \( P(y_1, y_2, y_3, \ldots | x_1, x_2, x_3, \ldots) \)...

- ...which in turn requires “expectations of marginals,” \( P(y_1 | x_1, x_2, x_3, \ldots) \)

- But, getting marginal distributions by sampling can be inefficient due to large sample space.

- Alternative: Perceptron. Approximate gradient from difference between true output and model’s predicted best output.
Parameter Estimation in Large State Spaces

- Most methods require calculating gradient of log-likelihood, \( P(y_1, y_2, y_3, \ldots \mid x_1, x_2, x_3, \ldots) \). …

- …which in turn requires “expectations of marginals,” \( P(y_1 \mid x_1, x_2, x_3, \ldots) \).

- But, getting marginal distributions by sampling can be inefficient due to large sample space.

- Alternative: Perceptron. Approximate gradient from difference between true output and model’s predicted best output.

- But, even finding model’s predicted best output is expensive.
Parameter Estimation in Large State Spaces

- Most methods require calculating gradient of log-likelihood, \( P(y_1, y_2, y_3, \ldots \mid x_1, x_2, x_3, \ldots) \)

- ...which in turn requires “expectations of marginals,” \( P(y_1 \mid x_1, x_2, x_3, \ldots) \)

- But, getting marginal distributions by sampling can be inefficient due to large sample space.

- Alternative: Perceptron. Approximate gradient from difference between true output and model’s predicted best output.

- But, even finding model’s predicted best output is expensive.

- We propose:
  - SampleRank [Culotta, Wick, Hall, McCallum, HLT 2007]
  - Training with Reinforcement Learning [Wick, Rohanimanesh, Singh, McCallum, NIPS 2009]
Most methods require calculating gradient of log-likelihood, \( P(y_1, y_2, y_3, \ldots | x_1, x_2, x_3, \ldots) \)

...which in turn requires “expectations of marginals,”
\( P(y_1 | x_1, x_2, x_3, \ldots) \)

But, getting marginal distributions by sampling can be inefficient due to large sample space.

Alternative: Perceptron. Approximate gradient from difference between true output and model’s predicted best output.

But, even finding model’s predicted best output is expensive.

We propose:

- **SampleRank** [Culotta, Wick, Hall, McCallum, HLT 2007]
- **Training with Reinforcement Learning** [Wick, Rohanimanesh, Singh, McCallum, NIPS 2009]

Learn to rank intermediate solutions:
\( P(y_1=1, y_2=0, y_3=1, \ldots | \ldots) > P(y_1=0, y_2=0, y_3=1, \ldots | \ldots) \)
Ranking vs Classification Training

• Instead of training

\[
\begin{align*}
\text{[Powell, Mr. Powell, he]} & \rightarrow \text{YES} \\
\text{[Powell, Mr. Powell, she]} & \rightarrow \text{NO}
\end{align*}
\]

• ...Rather...

\[
\begin{align*}
\text{[Powell, Mr. Powell, he]} & \succ [\text{Powell, Mr. Powell, she]}
\end{align*}
\]

• In general, higher-ranked example may contain errors

\[
\begin{align*}
\text{[Powell, Mr. Powell, George, he]} & \succ [\text{Powell, Mr. Powell, George, she]}
\end{align*}
\]
Ranking Intermediate Solutions

Example

1.
Ranking Intermediate Solutions

Example

1. 2.

$\Delta \text{Model} = -23$
$\Delta \text{Truth} = -0.2$
Ranking Intermediate Solutions

Example

1. 2. 3.

$\Delta \text{Model} = -23$
$\Delta \text{Truth} = -0.2$

$\Delta \text{Model} = 10$
$\Delta \text{Truth} = -0.1$
Ranking Intermediate Solutions

Example

1. \( \Delta \text{Model} = -23 \)
   \( \Delta \text{Truth} = -0.2 \)

2. \( \Delta \text{Model} = 10 \)
   \( \Delta \text{Truth} = -0.1 \)

UPDATE
Ranking Intermediate Solutions

Example

1. Δ Model = -23
   Δ Truth = -0.2

2. Δ Model = 10
   Δ Truth = -0.1

3. Δ Model = -10
   Δ Truth = -0.1

UPDATE
Ranking Intermediate Solutions

Example

1.  $\Delta \text{Model} = -23$
   $\Delta \text{Truth} = -0.2$

2.  $\Delta \text{Model} = 10$
   $\Delta \text{Truth} = -0.1$

3.  $\Delta \text{Model} = -10$
   $\Delta \text{Truth} = -0.1$

4.  $\Delta \text{Model} = -3$
   $\Delta \text{Truth} = 0.3$

UPDATE
Ranking Intermediate Solutions

Example

1. \[ \Delta \text{Model} = -23 \]
\[ \Delta \text{Truth} = -0.2 \]

2. \[ \Delta \text{Model} = 10 \]
\[ \Delta \text{Truth} = -0.1 \]

3. \[ \Delta \text{Model} = -10 \]
\[ \Delta \text{Truth} = -0.1 \]

4. \[ \Delta \text{Model} = -3 \]
\[ \Delta \text{Truth} = 0.3 \]
Ranking Intermediate Solutions

Example

1. \[ \Delta \text{Model} = -23 \]
   \[ \Delta \text{Truth} = -0.2 \]

2. \[ \Delta \text{Model} = 10 \]
   \[ \Delta \text{Truth} = -0.1 \]

3. \[ \Delta \text{Model} = -10 \]
   \[ \Delta \text{Truth} = -0.1 \]

4. \[ \Delta \text{Model} = -3 \]
   \[ \Delta \text{Truth} = 0.3 \]

• More constrained than Maximum Likelihood: Parameters must correctly rank \textit{incorrect} solutions!
More constrained than Maximum Likelihood:
Parameters must correctly rank _incorrect_ solutions!

SampleRank: Proof of convergence under Marginal Separability
**Ranking Intermediate Solutions Example**

1. \( \Delta \text{Model} = -23 \)
   \( \Delta \text{Truth} = -0.2 \)

2. \( \Delta \text{Model} = 10 \)
   \( \Delta \text{Truth} = -0.1 \)

3. \( \Delta \text{Model} = -10 \)
   \( \Delta \text{Truth} = -0.1 \)

4. \( \Delta \text{Model} = -3 \)
   \( \Delta \text{Truth} = 0.3 \)

- More constrained than Maximum Likelihood: Parameters must correctly rank *incorrect* solutions!

- SampleRank: Proof of convergence under Marginal Separability

- Reinforcement Learning: convergence not guaranteed but better results in practice
Ranking Intermediate Solutions

Example

1. \[ \Delta \text{Model} = -23 \]
   \[ \Delta \text{Truth} = -0.2 \]

2. \[ \Delta \text{Model} = 10 \]
   \[ \Delta \text{Truth} = -0.1 \]

3. \[ \Delta \text{Model} = -10 \]
   \[ \Delta \text{Truth} = -0.1 \]

4. \[ \Delta \text{Model} = -3 \]
   \[ \Delta \text{Truth} = 0.3 \]

- More constrained than Maximum Likelihood: Parameters must correctly rank *incorrect* solutions!
- SampleRank: Proof of convergence under Marginal Separability
- Reinforcement Learning: convergence not guaranteed but better results in practice
- *Much* faster to train
Escape from Local Minima

- Interesting objective functions like F1 are not convex
- Learn a policy to pass through bad clusterings on the way to good ones
## Experiment: Aligning Text w/DB

<table>
<thead>
<tr>
<th>Method</th>
<th>F1</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>89.9</td>
<td>100</td>
<td>81.5</td>
</tr>
<tr>
<td>Contrastive Divergence</td>
<td>76.9</td>
<td>78.0</td>
<td>57.0</td>
</tr>
<tr>
<td>SampleRank</td>
<td>92.0</td>
<td>88.9</td>
<td>76.3</td>
</tr>
<tr>
<td>Reinf. Learn.</td>
<td>94.3</td>
<td>96.1</td>
<td>92.6</td>
</tr>
</tbody>
</table>
FACTORIE So Far

• Factor graphs,

• ...scalable
  – factors created on demand, only score diffs

• ...with imperative hooks
  – jump function, override variable.set() for coordination
  – model structure
  – neighbor variables $\rightarrow$ sufficient statistics

• ...discriminative
  – efficient online training by local updates

• Combine declarative & procedural knowledge
Ongoing Work

• FACTORIE public release: early Dec.

• Efficient combinations of sampling and message passing

• Inference over variables of interest (e.g., PAR predicates) while marginalizing out others (e.g., syntax, pragmatics)

• Joint inference for morpho-syntax, semantics, and pragmatics
Thanks
Outline

- Motivate Joint Inference
- Brief introduction to Conditional Random Fields
- Example of Big, Hairy Conditional Random Field
  - Partition-wise Co-reference Resolution, Metropolis-Hastings
- Parameter Estimation with Sample Rank
- Example of Big, Hairy Joint Inference
  - Information Integration
- MCMC with Reinforcement Learning
- Probabilistic Programming with Factorie
  - Declarative + Imperative
Metropolis-Hastings

Given factor graph with target variables $y$ and observed $x$

$$P(y|x) = \frac{1}{Z_x} \prod_{y^i \in \mathcal{F}} \psi(x, y^i)$$

$\mathcal{F}$ feasible region defined by deterministic constraints
e.g. clustering, parse-tree projectivity.

$q$ proposal distribution

$$q(y'|y) : \mathcal{F} \times \mathcal{F} \rightarrow [0, 1]$$

1. Begin with some initial configuration $y_0 \in \mathcal{F}$
2. For $i=1,2,3,\ldots$ draw a local modification $y' \in \mathcal{F}$ from $q$
3. Probabilistically accept jump as Bernoulli draw with param $\alpha$

$$\alpha = \min \left( 1, \frac{p(y') q(y'|y')}{p(y) q(y'|y)} \right)$$

Can do MAP inference with decreasing temperature on ratio of $p(y)$'s
M-H Natural Efficiencies

1. Partition function cancels

\[
\frac{p(y')}{p(y)} = \frac{p(Y = y'|x; \theta)}{p(Y = y|x; \theta)} = \frac{\frac{1}{Z_x} \prod_{y^i \in y'} \psi(x, y^i)}{\frac{1}{Z_x} \prod_{y \in y} \psi(x, y^i)} = \frac{\prod_{y^i \in y'} \psi(x, y^i)}{\prod_{y \in y} \psi(x, y^i)}
\]

2. Unchanged factors cancel

\[
= \frac{\prod_{y^i \in y'} \psi(x, y^i)}{\prod_{y \in y} \psi(x, y^i)} = \frac{\left(\prod_{y^i \in \delta_y} \psi(x, y^i)\right)}{\left(\prod_{y \in y/\delta_y} \psi(x, y^i)\right)} \left(\prod_{y^i \in \delta_y} \psi(x, y^i)\right) \left(\prod_{y \in y/\delta_y} \psi(x, y^i)\right)
\]

How to learn parameters

\[\theta \propto p(Y = y|x; \theta)\]
Parameter Update Derivation

Given “proximity to truth” function

\[ F : \mathcal{F} \rightarrow \mathbb{R} \]

Let \( y' \) be in the neighborhood of \( y \) according to proposal manifold

\[
p(y'|x) > p(y|x) \iff F(y') > F(y)
\]

\[
\frac{p(y'|x)}{p(y|x)} > 1 \iff F(y') > F(y)
\]

\[
\log \left( \frac{p(y'|x)}{p(y|x)} \right) > 0 \iff F(y') > F(y)
\]

\[
\log p(y'|x) - \log p(y|x) > 0 \iff F(y') > F(y)
\]
Parameter Update Derivation

\[
\log p(y'|x) - \log p(y|x) = \log \prod_{y''i \in \delta'_y} \psi(x, y''i) - \log \prod_{y^i \in \delta_y} \psi(x, y^i) \\
= \sum_{y''i \in \delta'_y} \log \psi(x, y''i) - \sum_{y^i \in \delta_y} \log \psi(x, y^i) \\
= \sum_{y''i \in \delta'_y} \theta \cdot \phi - \sum_{y^i \in \delta_y} \theta \cdot \phi \\
= \theta \cdot \sum_{y''i \in \delta'_y} \phi - \theta \cdot \sum_{y^i \in \delta_y} \theta \phi \\
= \theta \cdot \phi_{y',y}
\]

We want \( \theta \cdot \phi_{y',y} > 0 \) ... use \( \phi_{y',y} \) to update