Machine Learning applied to Natural Language Processing

Lluís Màrquez
TALP Research Center, Software Department
Technical University of Catalonia

Advanced Methods for Corpus Processing
UPV/EHU Master: Analysis and Processing of Language

Donostia, January 18-19, 2010

Document composed using: **pdflatex**, **ppower4**, **xfig** (with multi meta post format), **mpost**

Course Overview

- Introduction and Motivation **(10%)**
- Machine Learning for supervised classification **(35%)**

Course Overview

- Introduction and Motivation **(10%)**
- Machine Learning for supervised classification **(35%)**
  - ⋆ Machine learning algorithms
    - ∗ Decision trees
    - ∗ AdaBoost
    - ∗ Linear classifiers
    - ∗ Support Vector Machines
  - ⋆ Applications
    - ∗ POS tagging
    - ∗ Document categorization

Course Overview

- Introduction and Motivation **(10%)**
- Machine Learning for supervised classification **(35%)**
  - **Machine learning algorithms**
    - **Decision trees**
    - **AdaBoost**
    - **Linear classifiers**
    - **Support Vector Machines**
  - Applications
    - POS tagging
    - Document categorization

Course Overview

- Introduction and Motivation **(10%)**
- Machine Learning for supervised classification **(35%)**
- Statistical Learning for structured NLP **(35%)**

Course Overview

- Introduction and Motivation **(10%)**
- Machine Learning for supervised classification **(35%)**
- Statistical Learning for structured NLP **(35%)**
  - ⋆ Learning Paradigms
    - ∗ Generative Models
    - ∗ Learning and Inference
    - ∗ Re-ranking
    - ∗ Global Linear Models
  - ⋆ Applications
    - ∗ Named entity recognition and relation extraction
    - ∗ Clause splitting
    - ∗ Syntactic parsing

Course Overview

- Introduction and Motivation **(10%)**
- Machine Learning for supervised classification **(35%)**
- Statistical Learning for structured NLP **(35%)**
  - Learning Paradigms
    - Generative Models
    - **Learning and Inference**
    - **Re-ranking**
    - Global Linear Models (**Structure Perceptron**)
  - Applications
    - Named entity recognition and relation extraction
    - **Clause splitting**
    - Syntactic parsing

Course Overview

- Introduction and Motivation **(∼10%, 40m)**
- Machine Learning for supervised classification **(∼35%, 2h20m)**
- Statistical Learning for structured NLP **(∼35%, 2h20m)**
- A case study: Semantic Role Labeling **(∼20%, 1h30m)**

Course Overview

- **Introduction and Motivation**
  - ⋆ NLP tasks and Learning
  - ⋆ NLP tools in the Web
- Machine Learning for supervised classification
- Statistical Learning for structured NLP
- A case study: Semantic Role Labeling

Natural Language Processing Applications

- Typical Applications (among others):
  - Machine Translation
  - *Intelligent* Information Retrieval and document managment
  - Information Extraction
  - Question&Answering
  - Document Summarization (multidocument, multilingual)
  - Dialog Systems

Natural Language Processing Applications

- Typical Applications (among others):
  - Machine Translation
  - *Intelligent* Information Retrieval and document managment
  - Information Extraction
  - Question&Answering
  - Document Summarization (multidocument, multilingual)
  - Dialog Systems
- Different levels of linguistic knowledge/comprehension are required

Natural Language Processing Applications

- Typical Applications (among others):
  - Machine Translation
  - *Intelligent* Information Retrieval and document managment
  - Information Extraction
  - Question&Answering
  - Document Summarization (multidocument, multilingual)
  - Dialog Systems
- Different levels of linguistic knowledge/comprehension are required
- They need to resolve a number of basic subproblems

Natural Language Processing Problems$_{(1)}$

Part–of–Speech Tagging

The San Francisco Examiner issued a special edition around noon yesterday that was filled entirely with earthquake new and information.

Natural Language Processing Problems$_{(1)}$

Part–of–Speech Tagging

The_DT San_NNP Francisco_NNP Examiner_NNP issued_VBD a_DT special_JJ edition_NN around_IN noon_NN yesterday_NN that_WDT was_VBD filled_VBN entirely_RB with_IN earthquake_NN news_NN and_CC information_NN ._.

Natural Language Processing Problems$_{(1)}$

Part–of–Speech Tagging

The_DT San_NNP Francisco_NNP Examiner_NNP issued_VBD a_DT special_JJ edition_NN around_IN noon_NN yesterday_NN that_WDT was_VBD filled_VBN entirely_RB with_IN earthquake_NN news_NN and_CC information_NN ._.

POS tagging is a pure sequential labeling problem

Natural Language Processing Problems$_{(1)}$

Part–of–Speech Tagging

The_DT San_NNP Francisco_NNP Examiner_NNP issued_VBD a_DT special_JJ edition_NN around_IN noon_NN yesterday_NN that_WDT was_VBD filled_VBN entirely_RB with_IN earthquake_NN news_NN and_CC information_NN ._.

$\boxed{\text{POS tagging is a pure sequential labeling problem}}$

(sequential learning paradigm)

Natural Language Processing Problems$_{(1)}$

Part–of–Speech Tagging

The_DT San_NNP Francisco_NNP Examiner_NNP issued_VBD a_DT special_JJ edition_NN around_IN noon_NN yesterday_NN that_WDT was_VBD filled_VBN entirely_RB with_IN earthquake_NN news_NN and_CC information_NN ._.

POS tagging is a pure sequential labeling problem

(sequential learning paradigm)

But... are really words ambiguous with respect to POS?

Natural Language Processing Problems$_{(1)}$

Part–of–Speech Tagging

The_DT San_NNP Francisco_NNP Examiner_NNP issued_VBD a_DT special_JJ edition_NN around_IN noon_NN yesterday_NN that_WDT was_VBD filled_VBN entirely_RB with_IN earthquake_NN news_NN and_CC information_NN ._.

But... are really words ambiguous with respect to POS?

Let's take a look at a free on-line demo: **FreeLing**
http://garraf.epsevg.upc.es/freeling/demo.php

Natural Language Processing Problems$_{(2)}$

Syntactic Analysis (Parsing)

Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.

Natural Language Processing Problems$_{(2)}$

Syntactic Analysis (Parsing)

Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.

```
((S (NP-SBJ
        (NP (NNP Pierre) (NNP Vinken) )
        (, ,)
        (ADJP
            (NP (CD 61) (NNS years) )
            (JJ old) )
        (, ,) )
    (VP (MD will)
        (VP (VB join)
            (NP (DT the) (NN board) )
            (PP-CLR (IN as)
                (NP (DT a) (JJ nonexecutive) (NN director) ))
            (NP-TMP (NNP Nov.) (CD 29) )))
    (. .) ))
```

Natural Language Processing Problems$_{(3)}$

Shallow Parsing (Chunking)

He reckons the current account deficit will narrow to only 1.8 billion in September.

Natural Language Processing Problems$_{(3)}$

Shallow Parsing (Chunking)

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only 1.8 billion] [PP in] [NP September] .

Natural Language Processing Problems$_{(3)}$

Shallow Parsing (Chunking)

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only 1.8 billion] [PP in] [NP September] .

Chunking is a sequential phrase recognition task

Natural Language Processing Problems$_{(3)}$

Shallow Parsing (Chunking)

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only 1.8 billion] [PP in] [NP September] .

$\boxed{\text{Chunking is a sequential phrase recognition task}}$

It can be seen as a sequential labeling problem (B-I-O encoding)

He_B-NP reckons_B-VP the_B-NP current_I-NP account_I-NP deficit_I-NP will_B-VP narrow_I-VP to_B-PP only_B-NP 1.8_I-NP billion_I-NP in_B-PP September_B-NP ._O

Natural Language Processing Problems$_{(3)}$

Shallow Parsing (Chunking)

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only 1.8 billion] [PP in] [NP September] .

Chunking is a sequential phrase recognition task

It can be seen as a sequential labeling problem (B-I-O encoding)

He_B-NP reckons_B-VP the_B-NP current_I-NP account_I-NP deficit_I-NP will_B-VP narrow_I-VP to_B-PP only_B-NP 1.8_I-NP billion_I-NP in_B-PP September_B-NP ._O

this is simple, but...

Natural Language Processing Problems$_{(4)}$

Clause splitting (partial parsing)

The deregulation of railroads and trucking companies that began in 1980 enabled shippers to bargain for transportation.

Natural Language Processing Problems$_{(4)}$

Clause splitting (partial parsing)

(S The deregulation of railroads and trucking companies (SBAR that (S began in 1980)) enabled (S shippers to bargain for transportation) .)

Natural Language Processing Problems$_{(4)}$

Clause splitting (partial parsing)

(S The deregulation of railroads and trucking companies
(SBAR that
(S began in 1980))
enabled
(S shippers to bargain for transportation)
.)

Clauses may embed: they form a hierarchy

Natural Language Processing Problems$_{(4)}$

Clause splitting (partial parsing)

(S The deregulation of railroads and trucking companies
(SBAR that
(S began in 1980))
enabled
(S shippers to bargain for transportation)
.)

Clauses may embed: they form a hierarchy

Clause splitting is a hierarchical prhase recognition problem

Natural Language Processing Problems$_{(4)}$

Clause splitting (partial parsing)

(S The deregulation of railroads and trucking companies
    (SBAR that
        (S began in 1980))
    enabled
    (S shippers to bargain for transportation)
    .)

Clauses may embed: they form a hierarchy

$\boxed{\text{Clause splitting is a hierarchical prhase recognition problem}}$

Not a good idea to treat it as a sequential problem...

Natural Language Processing Problems$_{(5)}$

Semantic Role Labeling (shallow semantic parsing)

He wouldn't accept anything of value from those he was writing about.

Natural Language Processing Problems$_{(5)}$

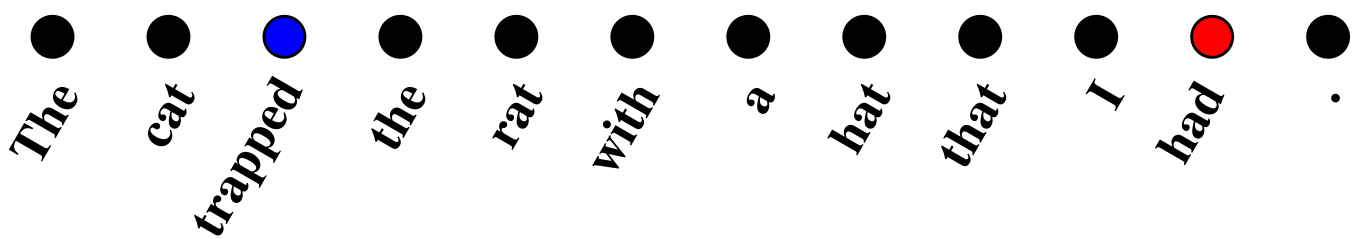
Semantic Role Labeling (shallow semantic parsing)

[$_{\mathrm{A}_0}$ He] [$_{\mathrm{AM\text{-}MOD}}$ would] [$_{\mathrm{AM\text{-}NEG}}$ n't] [$_{\mathrm{V}}$ **accept**] [$_{\mathrm{A}_1}$ anything of value] from [$_{\mathrm{A}_2}$ those he was writing about] .

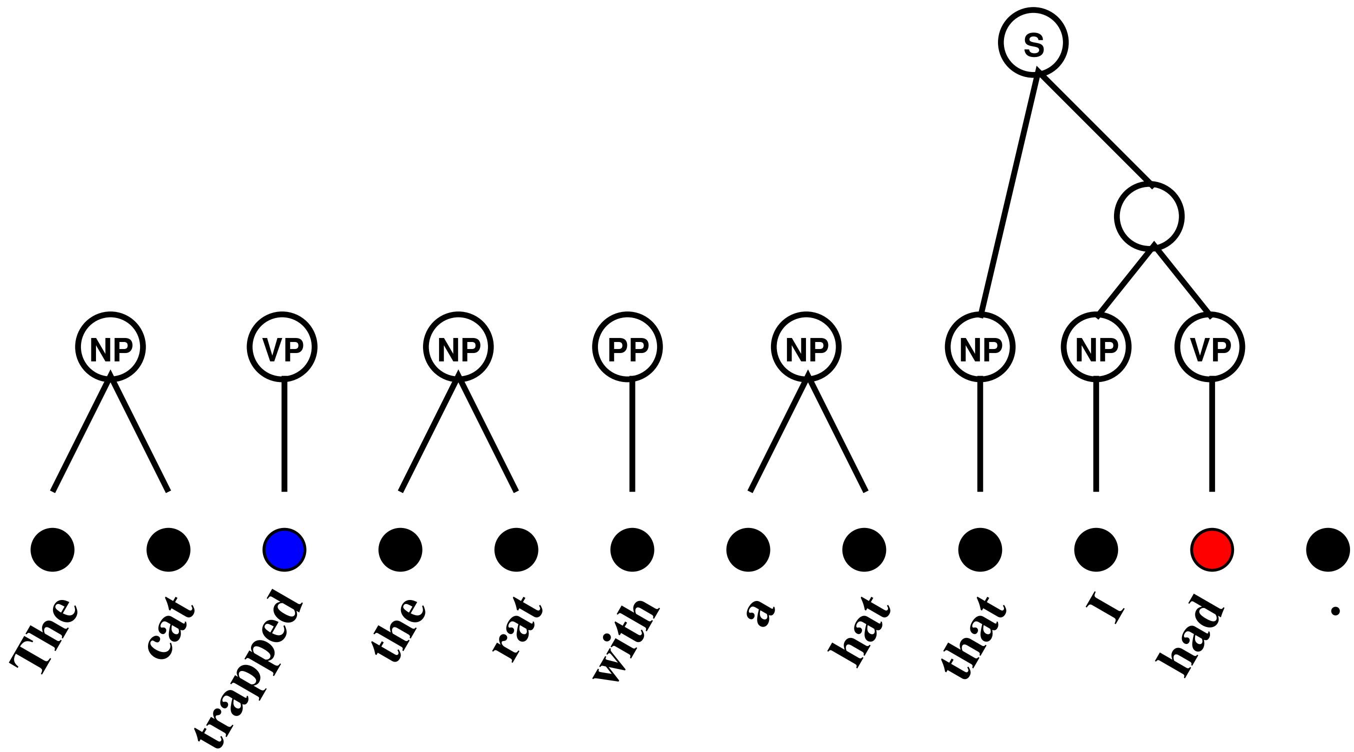
Roles for the predicate accept (PropBank Frames scheme):

V: verb; A_0: acceptor; A_1: thing accepted; A_2: accepted-from; A_3: attribute; AM-MOD: modal; AM-NEG: negation;

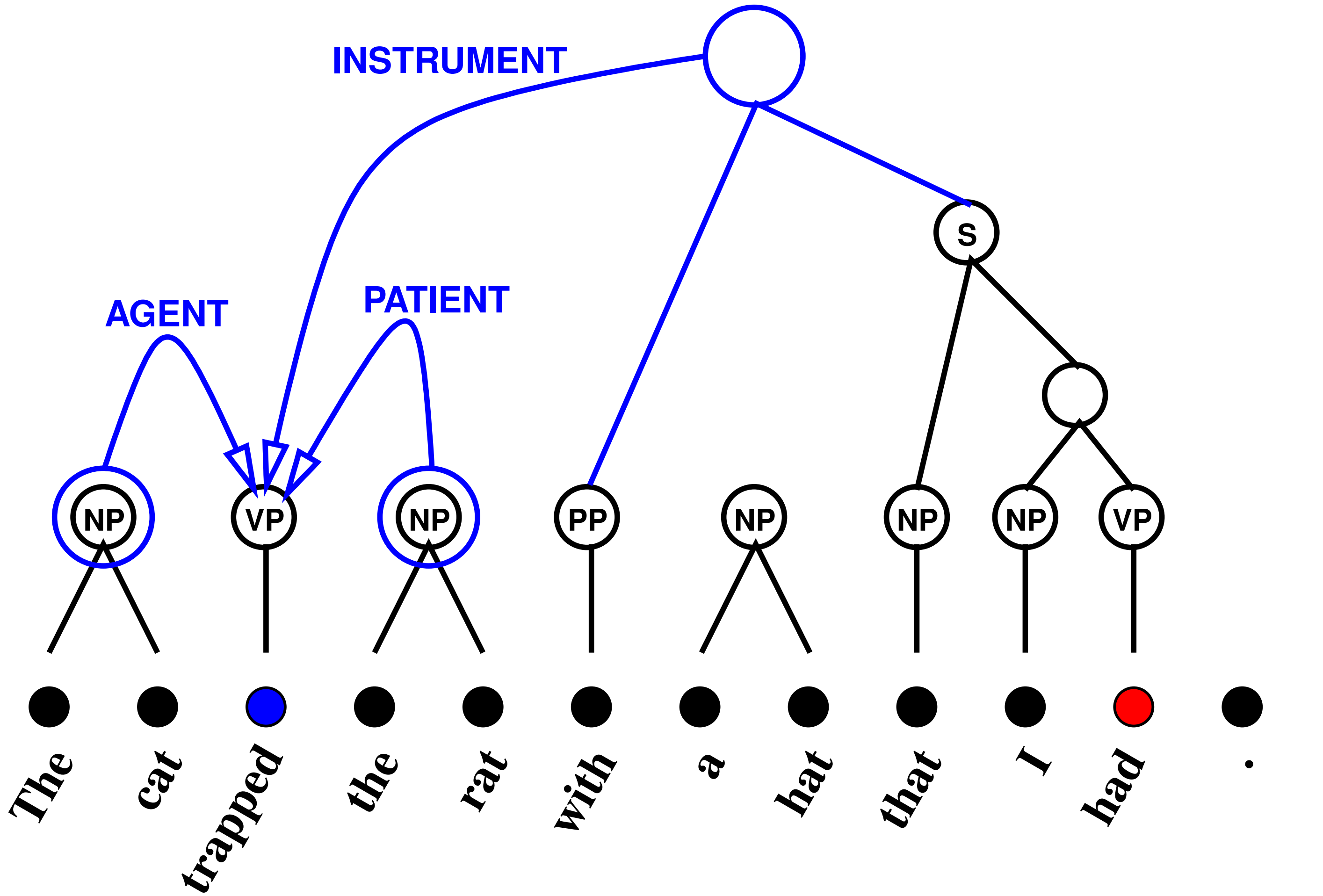
Natural Language Processing Problems$_{(5)}$



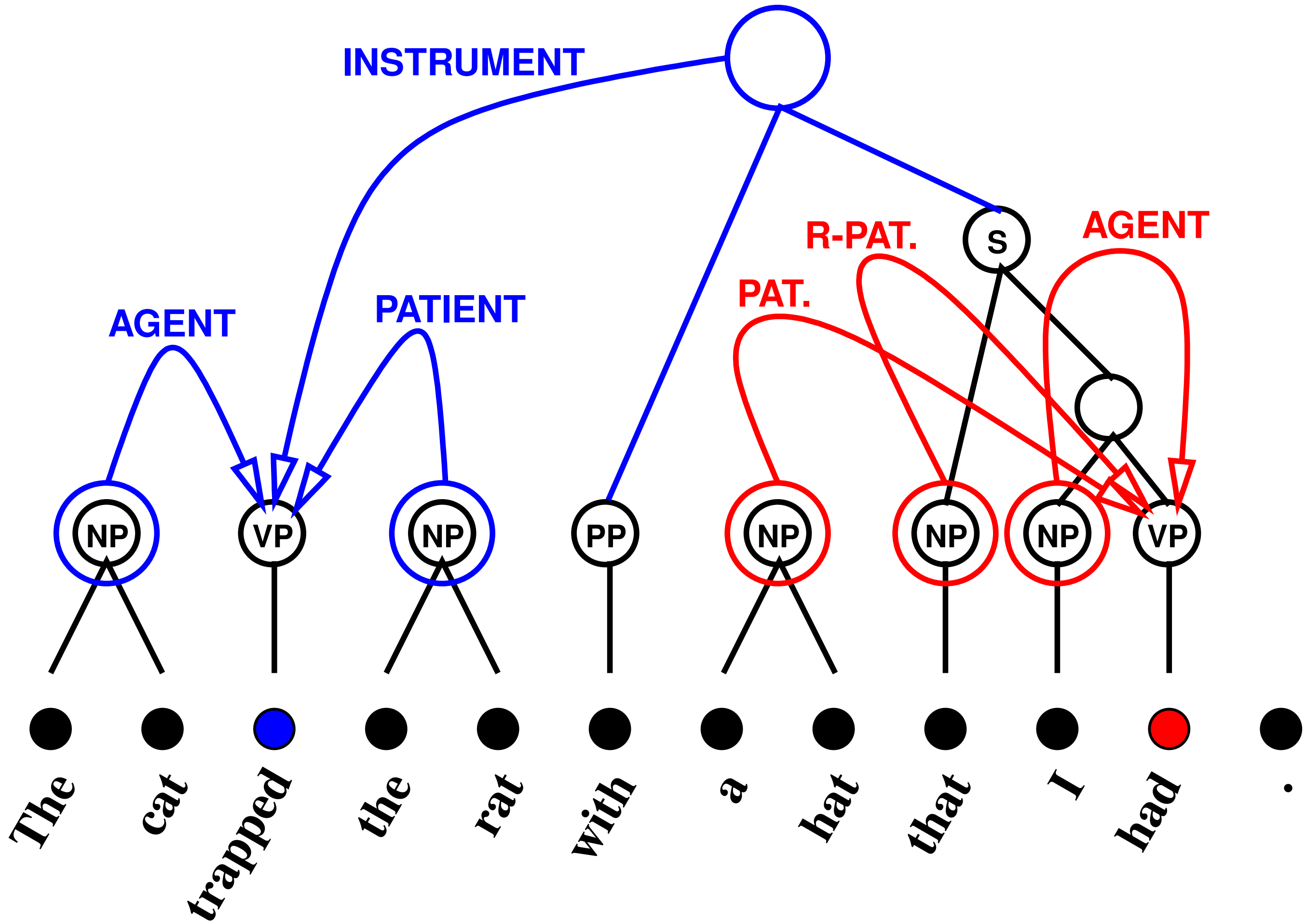
Natural Language Processing Problems$_{(5)}$



Natural Language Processing Problems$_{(5)}$



Natural Language Processing Problems$_{(5)}$



Natural Language Processing Problems$_{(6)}$

Named Entity Extraction ("semantic chunking")

Wolff, currently a journalist in Argentina, played with Del Bosque in the final years of the seventies in Real Madrid.

Natural Language Processing Problems$_{(6)}$

Named Entity Extraction ("semantic chunking")

[PER Wolff] , currently a journalist in [LOC Argentina] , played with [PER Del Bosque] in the final years of the seventies in [ORG Real Madrid] .

Natural Language Processing Problems$_{(6)}$

Named Entity Extraction ("semantic chunking")

[PER Wolff] , currently a journalist in [LOC Argentina] , played with [PER Del Bosque] in the final years of the seventies in [ORG Real Madrid] .

- Named Entities may be embedded
- NE tracing: variants and co-reference resolution
- Relations between entities: event extraction

Natural Language Processing Problems$_{(6)}$

Named Entities, relations, events, etc.

(example from the ACE corpus)

LOS ANGELES, April 18 (AFP)

Best−selling novelist and "Jurassic Park" creator Michael Crichton has agreed to pay his fourth wife 31 million dollars as part of their divorce settlement, court documents showed Friday.

Crichton, 60, is one of the world's wealthiest authors, and has had 12 of his novels made into major Hollywood movies.

The writer will retain the rights to his books and films, although he has agreed to split a raft of other possessions with Anne Marie, his wife of 13 years, according to documents filed in Los Angeles Superior Court.

Entity
Michael Crichton
PER−Individual−SPC

Entity
Anne−Marie
PER−Individual−SPC

Time
13 Years
Ending in 2003−04−18

Relation
PER−SOC−Family
Asserted
Past

Natural Language Processing Problems

Summary:

- Mapping from an input to an output structure
  - ⋆ The input structure is typically a sequence of words enriched with some linguistic information.
  - ⋆ Output structures are sequences, trees, graphs, etc.

Relation to Machine Learning

- 1980's resurgence of the empirical paradigm for NLP
- 1990's massive application of Machine Learning techniques
- **Important factor** (among others):
  - ⋆ $\boxed{\text{Ambiguity resolution can be directly casted as classification}}$

Relation to Machine Learning

- 1980's resurgence of the empirical paradigm for NLP
- 1990's massive application of Machine Learning techniques
- **Important factor** (among others):
  - ⋆ $\boxed{\text{Ambiguity resolution can be directly casted as classification}}$
- Thus, we know very well how to model and learn local decisions
- But... there is a big gap in between pure classification and structure learning/generation. Pure classification tasks don't really exist!

Relation to Machine Learning

- 1980's resurgence of the empirical paradigm for NLP
- 1990's massive application of Machine Learning techniques
- **Important factor** (among others):
  - ⋆ Ambiguity resolution can be directly casted as classification
- Thus, we know very well how to model and learn local decisions
- But... there is a big gap in between pure classification and structure learning/generation. Pure classification tasks don't really exist!
- Search is strongly related to the generation of the output structure (*decoding*, *inference*, etc.)

Main Challenges

...of the learning approach to structured NLP:

- Put learning at the level of the structure and design features accordingly
- Make **computationally efficient** learning and decoding algorithms
- Avoid locality: **global** (and coupled) learning and decoding
- Increase output complexity

Opportunities for the future

...of this new technology:

- Exploit linguistically rich features and complex dependencies
- Surpass the traditional NLP architecture of a pipeline of processors
- Approach multitask learning
- Design intermediate structures to learn, which are optimal for the global performance

Why applying Machine Learning?

- **Low cost** development of linguistic processors
- Language (quasi)independence: **reusability**
- Ability of **acquiring/discovering knowledge** from very large datasets
- **Assist manual development** of linguistic resources

On-line Demos in the Web

- FreeLing. Universitat Politècnica de Catalunya. Basic syntactic processing. Catalan and Spanish. http://garraf.epsevg.upc.es/freeling/demo.php

- CCG tools. University of Illinois at Urbana-Champaign. Multiple processors and applications. English. http://l2r.cs.uiuc.edu/∼cogcomp/demos.php

Course Overview

- Introduction and Motivation
  - ⋆ NLP tasks and Learning
  - ⋆ NLP tools in the Web
- **Machine Learning for supervised classification**
  - ⋆ Machine learning algorithms
    - ∗ Decision trees, AdaBoost, Linear classifiers,Support Vector Machines
  - ⋆ Applications
    - ∗ POS tagging, Document categorization
- Statistical Learning for structured NLP
- A case study: Semantic Role Labeling

Machine Learning for Supervised Classification

1. **Supervised Learning for Classification**: definitions and concepts
2. **Machine learning algorithms**: decision tree induction, boosting, linear classifiers, support vector machines
3. **Applications**: POS tagging, document categorization

- All three previous points are covered by the complementary slides on supervised learning and algorithms (in PowerPoint)

A brief note on features

- The learning algorithm deals with a representation of training examples x
- Feature codification function: $\Phi : \mathcal{X} \longrightarrow \mathbb{R}^n$
- $\Phi(x)$ is a vector of features, with values in $\mathbb{R}$.
- Basic feature codification for NLP local decision problems follow the sliding window approach: codification of the local context.

Extracting features: Sliding Window

?

. . . veí del carrer Santa Tecla de Girona , Josep . . .

Extracting features: Sliding Window

?

. . . veí del carrer Santa Tecla de Girona , Josep . . .

	−3	−2	−1	0	+1	+2	+3
Form	del	carrer	santa	tecla	de	girona	,
PoS	contr	n	adj	n	prep	n	,
Orto	min	min	Maj	Maj	min	Maj	punct
Prefix3	del	car	san	tec	de	gir	,
BIO	O	O	B				

Extracting features: Sliding Window

?

... veí del carrer Santa Tecla de Girona , Josep ...

	−3	−2	−1	0	+1	+2	+3
Form	del	carrer	santa	tecla	de	girona	,
PoS	contr	n	adj	n	prep	n	,
Orto	min	min	Maj	Maj	min	Maj	punct
Prefix3	del	car	san	tec	de	gir	,
BIO	O	O	B				

`form@-2=carrer`

Extracting features: Sliding Window

?

... veí del carrer Santa Tecla de Girona , Josep ...

	−3	−2	−1	0	+1	+2	+3
Form	del	carrer	santa	tecla	de	girona	,
PoS	contr	n	adj	n	prep	n	,
Orto	min	min	Maj	Maj	min	Maj	punct
Prefix3	del	car	san	tec	de	gir	,
BIO	O	O	B				

form@−2=carrer

pos@+1=prep

Extracting features: Sliding Window

?

... veí del carrer Santa Tecla de Girona , Josep ...

	−3	−2	−1	0	+1	+2	+3
Form	del	carrer	santa	tecla	de	girona	,
PoS	contr	n	adj	n	prep	n	,
Orto	min	min	Maj	Maj	min	Maj	punct
Prefix3	del	car	san	tec	de	gir	,
BIO	O	O	B				

form@−2=carrer

pos@+1=prep

orto@−1:0=MajMaj

Extracting features: Sliding Window

?

... veí del carrer Santa Tecla de Girona , Josep ...

	−3	−2	−1	0	+1	+2	+3
Form	del	carrer	santa	tecla	de	girona	,
PoS	contr	n	adj	n	prep	n	,
Orto	min	min	Maj	Maj	min	Maj	punct
Prefix3	del	car	san	tec	de	gir	,
BIO	O	O	B				

form@−2=carrer

orto@−1:0=MajMaj

pos@+1=prep

bio@−2:−1=OB

Snapshot of a codified example set$_{(1)}$

Confidence in the pound is widely expected to take another sharp dive ...

B-NP pos(+1):IN pos(-1):[OSB] pos(-1,0):[OSB]-NN pos(-1,0,+1):[OSB]-NN-IN pos(0):NN pos(0,+1):NN-IN w(+1):in w(-1):[OSB] w(-1,0):[OSB]-Confidence w(-1,0,+1):[OSB]-Confidence-in w(0):Confidence w(0,+1):Confidence-in
B-PP pos(+1):DT pos(-1):NN pos(-1,0):NN-IN pos(-1,0,+1):NN-IN-DT pos(0):IN pos(0,+1):IN-DT w(+1):the w(-1):Confidence w(-1,0):Confidence-in w(-1,0,+1):Confidence-in-the w(0):in w(0,+1):in-the
B-NP pos(+1):NN pos(-1):IN pos(-1,0):IN-DT pos(-1,0,+1):IN-DT-NN pos(0):DT pos(0,+1):DT-NN w(+1):pound w(-1):in w(-1,0):in-the w(-1,0,+1):in-the-pound w(0):the w(0,+1):the-pound
I-NP pos(+1):VBZ pos(-1):DT pos(-1,0):DT-NN pos(-1,0,+1):DT-NN-VBZ pos(0):NN pos(0,+1):NN-VBZ w(+1):is w(-1):the w(-1,0):the-pound w(-1,0,+1):the-pound-is w(0):pound w(0,+1):pound-is
B-VP pos(+1):RB pos(-1):NN pos(-1,0):NN-VBZ pos(-1,0,+1):NN-VBZ-RB pos(0):VBZ pos(0,+1):VBZ-RB w(+1):widely w(-1):pound w(-1,0):pound-is w(-1,0,+1):pound-is-widely w(0):is w(0,+1):is-widely
...

Snapshot of a codified example set$_{(2)}$

Numerical codification:

B-NP 1:1 6:1 31:1 33:1 41:1 84:1 559:1
B-PP 2:1 4:1 12:1 25:1 40:1 48:1 86:1 117:1 244:1
B-NP 3:1 5:1 11:1 27:1 29:1 47:1 83:1 85:1 243:1 2563:1 5741:1
I-NP 1:1 10:1 26:1 28:1 77:1 183:1 194:1 374:1 2714:1 5295:1
B-VP 2:1 68:1 76:1 185:1 192:1 420:1 774:1 2617:1 6501:1
I-VP 58:1 74:1 75:1 184:1 415:1 432:1 1214:1 1545:1 7769:1
I-VP 57:1 64:1 65:1 73:1 399:1 427:1 1251:1 2108:1 2827:1 6849:1
I-VP 56:1 63:1 67:1 72:1 100:1 396:1 547:1 1230:1 2022:1 2062:1 4471:1
I-VP 12:1 55:1 62:1 66:1 99:1 232:1 368:1 2102:1 2209:1 4234:1
B-NP 11:1 15:1 54:1 82:1 230:1 763:1 2056:1 2357:1 3362:1
...

Snapshot of a codified example set$_{(3)}$

Through a “dictionary” of features:

1 pos(0):NN 3208
2 pos(-1):NN 3206
3 pos(+1):NN 3168
4 pos(0):IN 2307
5 pos(-1):IN 2305
6 pos(+1):IN 2196
7 pos(0):NNP 2064
8 pos(-1):NNP 2062
9 pos(+1):NNP 1876
10 pos(-1):DT 1859
...
28 pos(-1,0):DT-NN 956
29 pos(0,+1):DT-NN 956
...

Course Overview

- Introduction and Motivation
  - ⋆ NLP tasks and Learning
  - ⋆ NLP tools in the Web
- Machine Learning for supervised classification
  - ⋆ Machine learning algorithms
  - ⋆ Applications
- **Statistical Learning for Structured NLP**
- A case study: Semantic Role Labeling

Supervised Machine Learning

- Given:
  - ⋆ A training set, with examples (x, y) where
    - ∗ $x \in \mathcal{X}$ could be sentences
    - ∗ $y \in \mathcal{Y}$ could be linguistic structures
    - ∗ We assume that the set was generated i.i.d. from an unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$
  - ⋆ An error function, or loss :
    $\mathrm{error}(y, \hat{y}) =$ cost of proposing $\hat{y}$ when the correct value was y
- Goal: learn a hypothesis

$$\mathrm{h} : \mathcal{X} \rightarrow \mathcal{Y}$$

that minimizes error on the entire distribution $\mathcal{D}$

Scenarios in Machine Learning

A general form of learning hypothesis:

$$\mathrm{h}(x) = \arg\max_{\hat{y} \in \mathcal{Y}} \mathrm{score}(x, \hat{y})$$

Depending on the output space $\mathcal{Y}$:

	Classes ($\mathcal{Y}$)	$\lvert\mathcal{Y}\rvert$	**enumeration of $\mathcal{Y}$**	**error**
Binary Classification	$\{+, -\}$	1	not needed	0-1
Multiclass Classification	$\{C_1, C_2, \ldots, C_m\}$	m	exhaustive	0-1
Structure Learning	all structures	exponential	not tractable	prec/rec on nodes

Structure Learning: Learning & Inference

- $\mathcal{Y}(x)$ is exponential on the size of x
- Not possible to exhaustively enumerate the output space
- Learning & Inference approach:
  - ⋆ Key Idea: decompose a structure into fragments
  - ⋆ Model: scores a structure by scoring its fragments
  - ⋆ Inference: search in $\mathcal{Y}(x)$ for the best scored solution for x
    - ∗ Build incrementally, instead of explore exhaustively
    - ∗ Use automata, grammars, . . . to build the solution
    - ∗ Use constraints to discard regions of $\mathcal{Y}(x)$

Course Overview

- Introduction and Motivation
  - ⋆ NLP tasks and Learning & NLP tools in the Web
- Machine Learning for supervised classification
  - ⋆ Machine learning algorithms & Applications
- **Statistical Learning for Structured NLP**
  - ⋆ Generative models
  - ⋆ The Learning and Inference paradigm
  - ⋆ Re-ranking candidate solutions
  - ⋆ Structure learning with global linear models
- A case study: Semantic Role Labeling

Generative Learning: Models

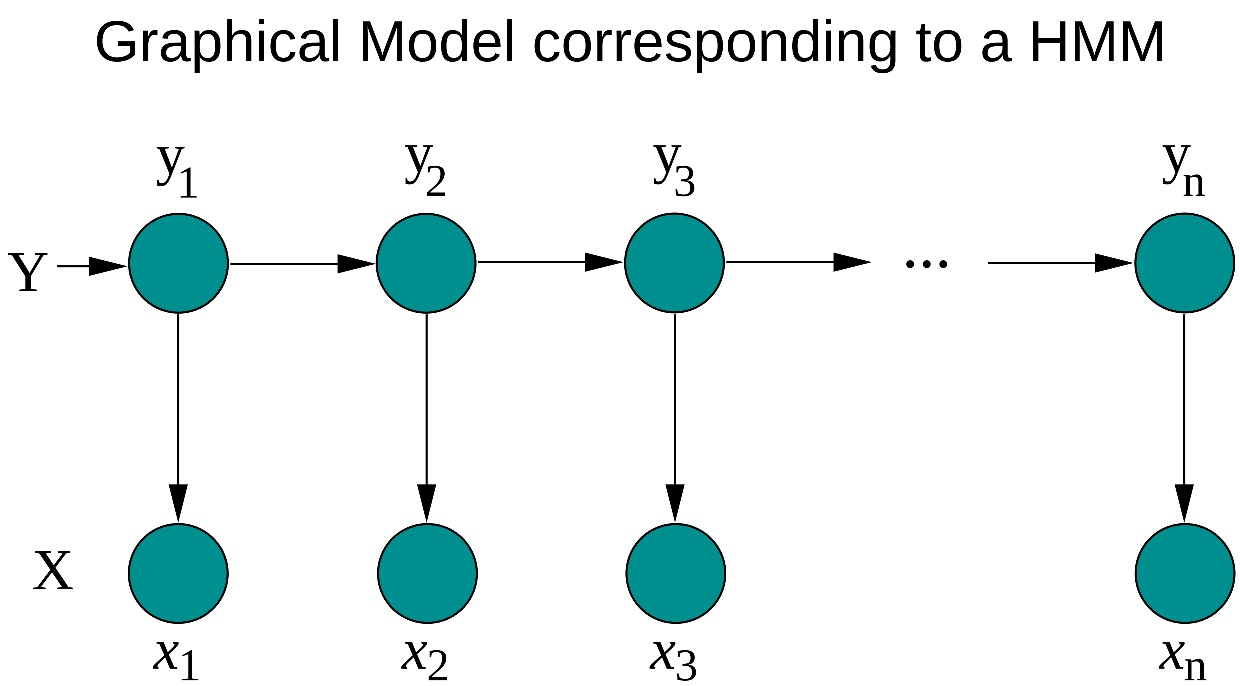
- Probabilistic models that define a joint probability distribution of the data: $p(\mathcal{X}, \mathcal{Y})$.

- The model is associated to a stochastic generation mechanism of the data, such as an automaton or grammar

- The graphical model underlying the generative mechanism is topologically sorted so as $\mathcal{X}$ variables never preceed $\mathcal{Y}$ variables

- For computational reasons one needs to define very restrictive simplifying assumptions on the generative process

Generative Learning: Models



- Paradigmatic models to recognize structure:
  - ⋆ Hidden Markov Models, e.g. **[Rabiner 89]**
  - ⋆ Probabilistic Context-Free Grammars, e.g. **[Collins 99]**

Generative Learning: Max-Likelihood Estimation

- Based on theory of probability and Bayesian learning:
- Training: via Maximum Likelihood, i.e., simple counts on the training data (very fast; but smoothing is needed)
- Inference Algorithms: efficient algorithms using dynamic programming e.g., Viterbi, CKY, etc.

Generative Models: HMM's

- Generation mechanism: probabilistic automaton with outputs
- Sequences of observations: $\{x_1, \ldots, x_n\}$ and states $\{y_1, \ldots, y_n\}$
- Assumptions: limited horizon (Markov order)
  x_i only depends on y_i

Generative Models: HMM's

- Generation mechanism: probabilistic automaton with outputs
- Sequences of observations: $\{x_1, \ldots, x_n\}$ and states $\{y_1, \ldots, y_n\}$
- Assumptions: limited horizon (Markov order)

  x_i only depends on y_i
- Objective function: $\arg\max_{y_1,\ldots,y_n} P(y_1, \ldots, y_n | x_1, \ldots, x_n) =$

$$\arg\max_{y_1,\ldots,y_n} \frac{P(x_1,\ldots,x_n|y_1,\ldots,y_n) \cdot P(y_1,\ldots,y_n)}{P(x_1,\ldots,x_n)} \approx$$

Generative Models: HMM's

- Generation mechanism: probabilistic automaton with outputs
- Sequences of observations: $\{x_1, \ldots, x_n\}$ and states $\{y_1, \ldots, y_n\}$
- Assumptions: limited horizon (Markov order)

  x_i only depends on y_i
- Objective function: $\arg\max_{y_1,\ldots,y_n} P(y_1, \ldots, y_n | x_1, \ldots, x_n) =$

$$\arg\max_{y_1,\ldots,y_n} \frac{P(x_1,\ldots,x_n|y_1,\ldots,y_n) \cdot P(y_1,\ldots,y_n)}{P(x_1,\ldots,x_n)} \approx$$

$$\boxed{\arg\max_{y_1,\ldots,y_n} \prod_{k=1}^{n} P(y_k|y_{k-2}, y_{k-1}) \cdot P(x_k|y_k)}$$

Generative models: HMM's

- $\boxed{\arg\max_{y_1,\ldots,y_n} \prod_{k=1}^{n} P(y_k|y_{k-2}, y_{k-1}) \cdot P(x_k|y_k)}$

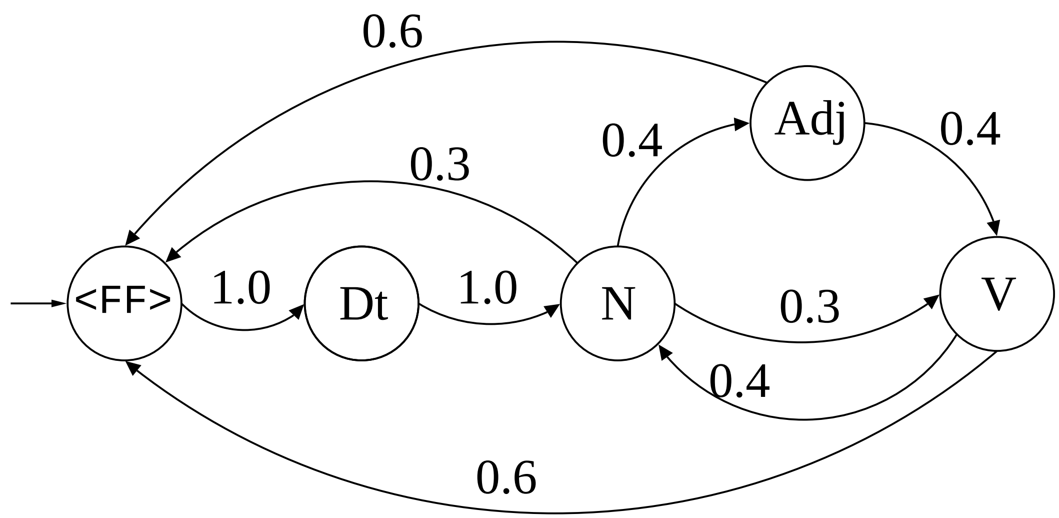
- We need to estimate the following probability distributions:
  - ⋆ emission probabilities: $P(x_k|y_k)$
  - ⋆ transition probabilities: $P(y_k|y_{k-2}, y_{k-1})$ (second order HMM)
  - ⋆ initial state probabilities: $P(y_1)$

- Viterbi algorithm allows to calculate the $\mathrm{arg\_max}$ in $O(n)$

- But there is a practically important constant factor: $MarkovOrder \times |States|$

Generative Models: HMM example

States and transition probabilities (first order HMM)



Emission probabilities	.	el	la	gato	niña	come	corre	pescado	fresco	pequeña	grande
<FF>	1.0										
Dt		0.6	0.4								
N				0.6	0.1			0.3			
V						0.7	0.3				
Adj									0.3	0.3	0.4

Generative Learning: example on NER

- IdentiFinder$^{\mathrm{TM}}$ **[Bikel, Schwartz and Weischedel 1999]**
- An HMM-based system for Named Entity Recognition, used at MUC conferences
- See complementary slides on IdentiFinder$^{\mathrm{TM}}$ (in PowerPoint)

Generative Learning: Pros and Cons

Advantages

- Flexibility to represent complex structures as generative processes
- Under certain simplifying assumptions:
  - ⋆ Simplicity of the training process: fast parameter estimation
  - ⋆ Very efficient decoding algorithms exist

Generative Learning: Pros and Cons

Problems

- Training/decoding on complex generative settings is not feasible
- Strong independence assumptions are needed: not necessary in accordance with data

Generative Learning: Pros and Cons

Problems

- Training/decoding on complex generative settings is not feasible
- Strong independence assumptions are needed: not necessary in accordance with data
- Difficult to use arbitrary feature representations
  - ⋆ Features are tied to the generation mechanism of the data
  - ⋆ Extending the feature dependendencies imply:
    - ∗ Severe sparsity problems (training is difficult)
    - ∗ Exact decoding may be computationally intractable

Generative Learning: Pros and Cons

Problems

- Training/decoding on complex generative settings is not feasible
- Strong independence assumptions are needed: not necessary in accordance with data
- Difficult to use arbitrary feature representations
  - ⋆ Features are tied to the generation mechanism of the data
  - ⋆ Extending the feature dependendencies imply:
    - ∗ Severe sparsity problems (training is difficult)
    - ∗ Exact decoding may be computationally intractable
  - ⋆ Feature specialization is possible but in a limited way

Alternative: Direct, Discriminative Learning

- ML methods that directly model $p(\mathcal{Y}|\mathcal{X})$
- Allow arbitrary representations; No assumptions needed
- Not necessarily probabilistic
- Mostly designed for classification (mostly binary)

Alternative: Direct, Discriminative Learning

- ML methods that directly model $p(\mathcal{Y}|\mathcal{X})$
- Allow arbitrary representations; No assumptions needed
- Not necessarily probabilistic
- Mostly designed for classification (mostly binary)
- A wide range of algorithms were used during the 80's and 90's:
  - ⋆ Maximum Entropy
  - ⋆ Decision Trees and Lists
  - ⋆ Memory-based
  - ⋆ Transformation-based
  - ⋆ Perceptron, Neural Nets
  - ⋆ AdaBoost
  - ⋆ Support Vector Machines
  - ⋆ ...

Alternative: Direct, Discriminative Learning

- ML methods that directly model $p(\mathcal{Y}|\mathcal{X})$
- Allow arbitrary representations; No assumptions needed
- Not necessarily probabilistic
- Mostly designed for classification (mostly binary)
- A wide range of algorithms were used during the 80's and 90's:
  - ⋆ Maximum Entropy
  - ⋆ Decision Trees and Lists
  - ⋆ Memory-based
  - ⋆ Transformation-based
  - ⋆ Perceptron, Neural Nets
  - ⋆ AdaBoost
  - ⋆ Support Vector Machines
  - ⋆ . . .
- Next, we study how to use these algorithms in structure learning

Course Overview

- Introduction and Motivation
  - ⋆ NLP tasks and Learning & NLP tools in the Web
- Machine Learning for supervised classification
  - ⋆ Machine learning algorithms & Applications
- **Statistical Learning for Structured NLP**
  - ⋆ Generative models
  - ⋆ The Learning and Inference paradigm
  - ⋆ Re-ranking candidate solutions
  - ⋆ Structure learning with global linear models
- A case study: Semantic Role Labeling

Learning and Inference: General Approach

- Transform the recognition problem into a chain of *simple* decisions:
  - ⋆ Segmentation Decisions:
    e.g., Open-Close, Begin-Inside-Outside, Shift-Reduce, etc.
  - ⋆ Labeling Decisions: made during segmentation or afterwards
  - ⋆ Decisions might use the output of earlier steps in the chain
- Set up an inference strategy:
  - ⋆ Decisions are applied in chain to build structure incrementally
  - ⋆ Exploration might be at different levels of amplitude:
    e.g., greedy, dynamic programming, beam search, etc.
- Learn a prediction function for each decision

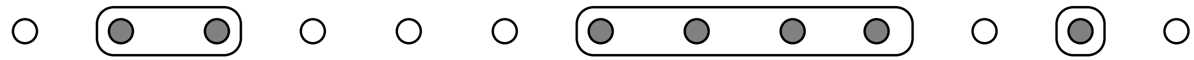
Learning & Inference: Local vs. Global Training

- Local training: each local function is trained separately, as a classifier (binary or multiclass)
  - ⋆ Good understanding on learning classifiers
  - ⋆ *but* local accuracies do not guarantee global accuracy
  - ⋆ *that is*, a local classification behavior might not be the optimal within inference
  - ⋆ *unless* local classifications are perfect

- Global training: train the recognizer as a composed function
  - ⋆ Local functions are trained dependently to optimize global accuracy. More on this later
  - ⋆ e.g., Linear models **[Collins 02,04]**, CRFs **[Lafferty et al. 01]**

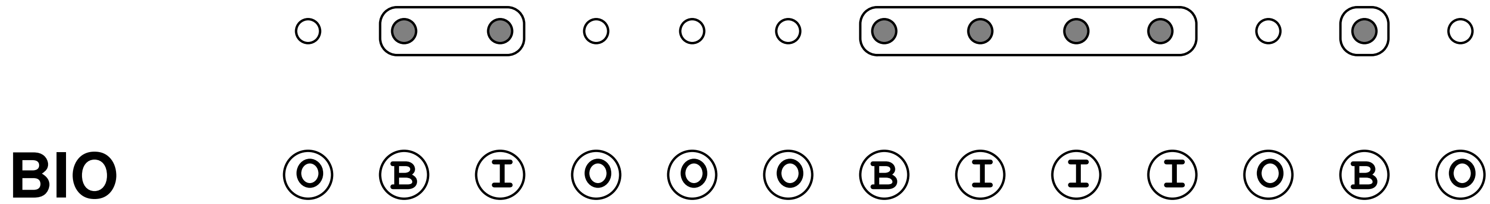
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



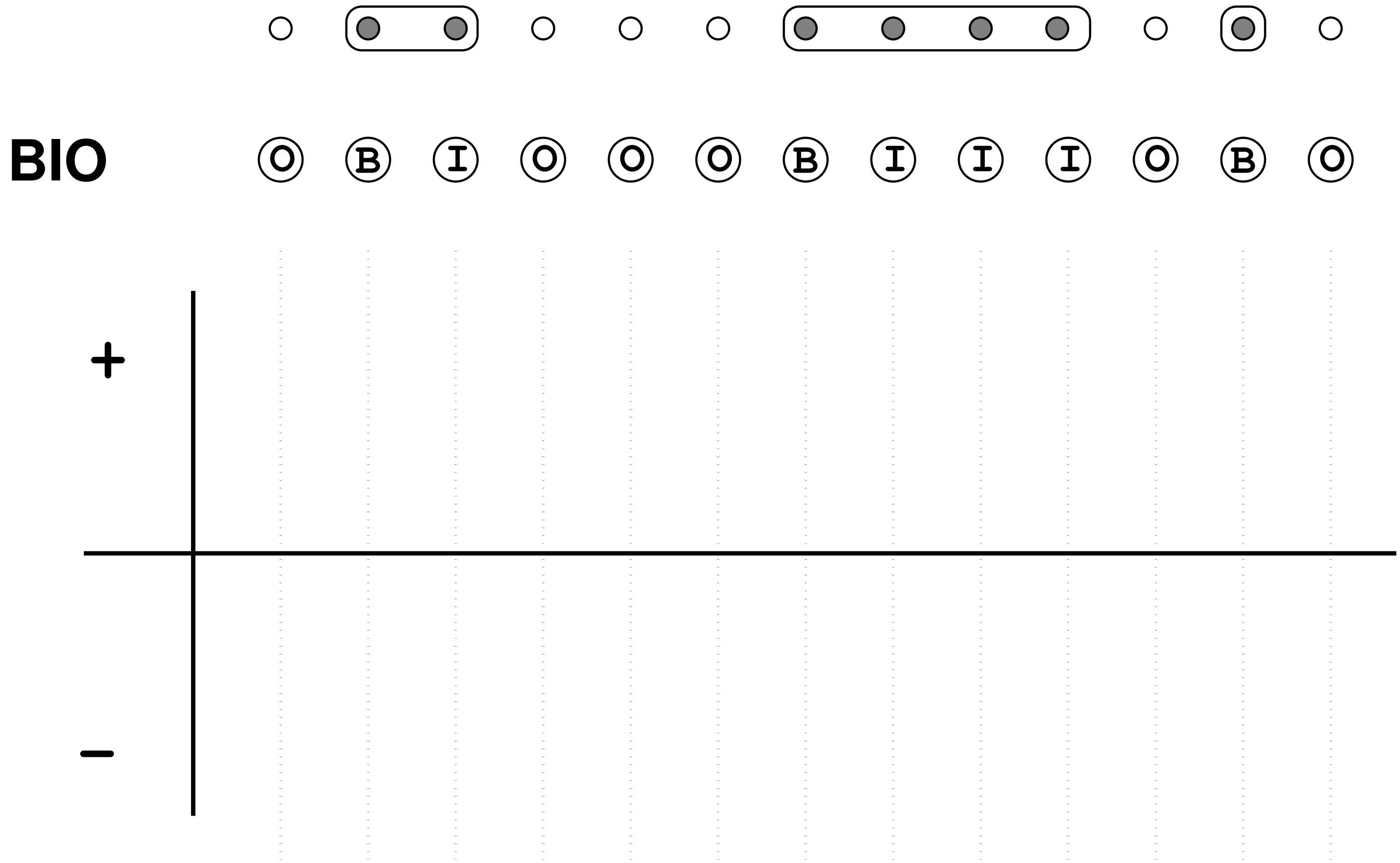
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



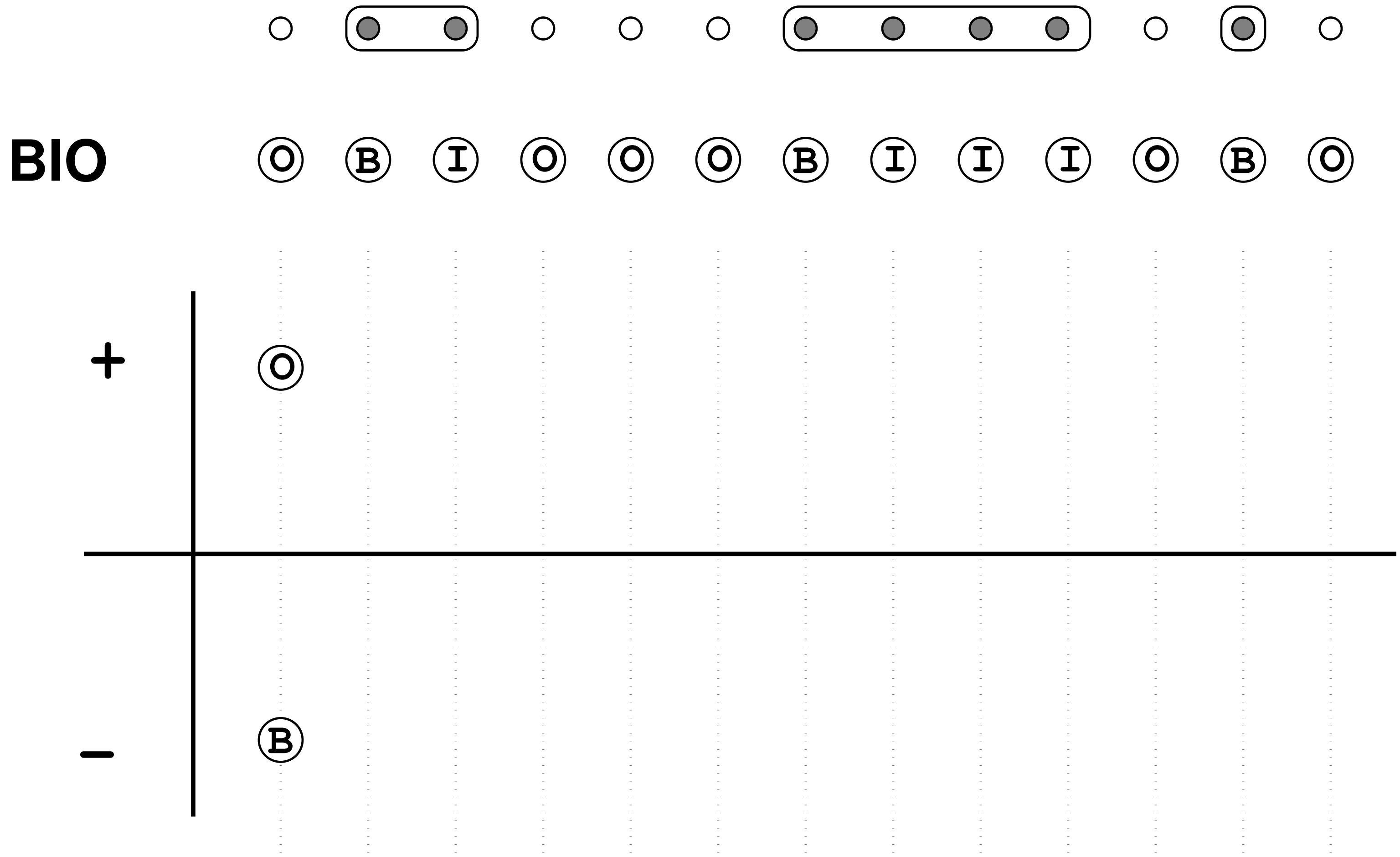
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



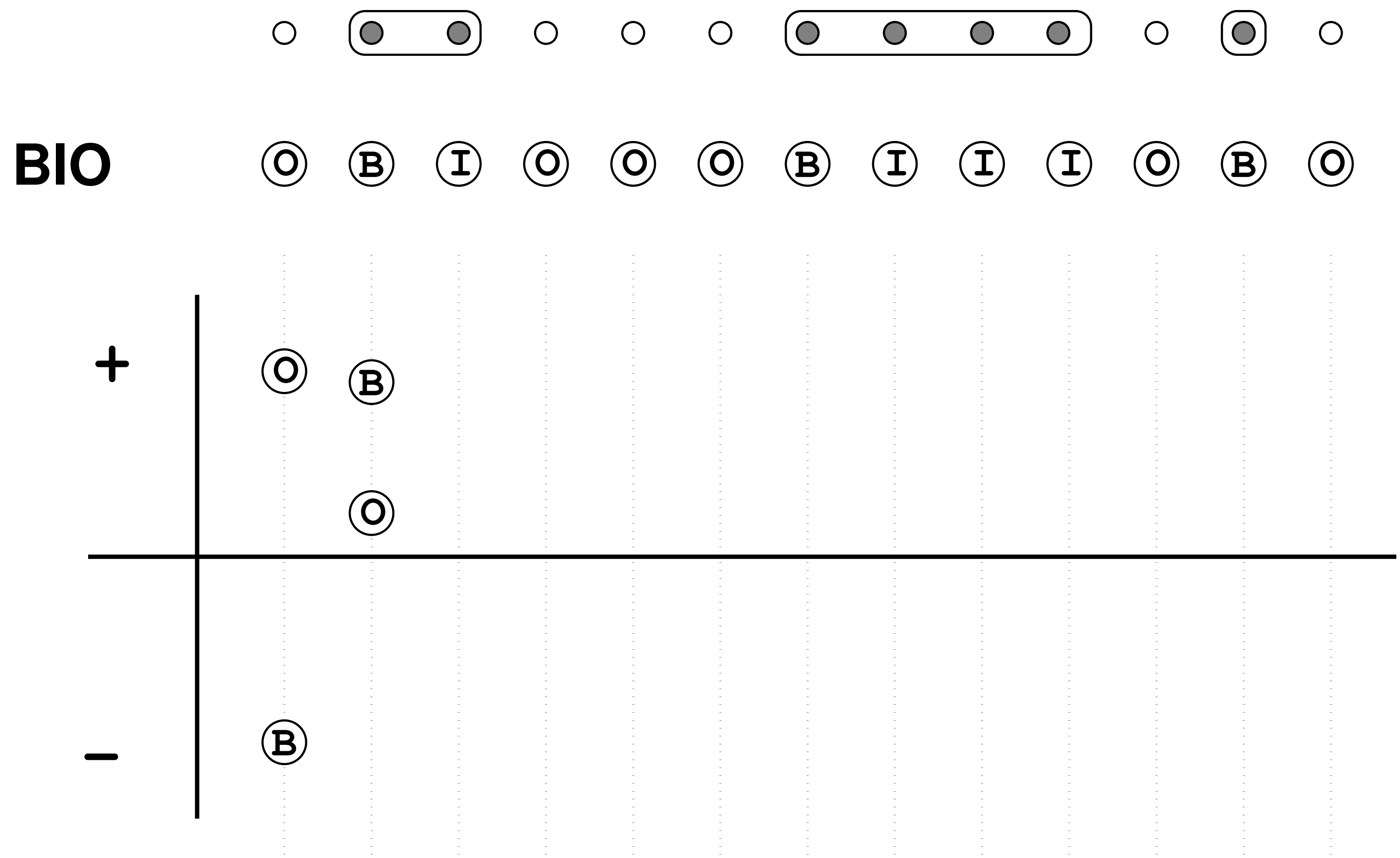
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



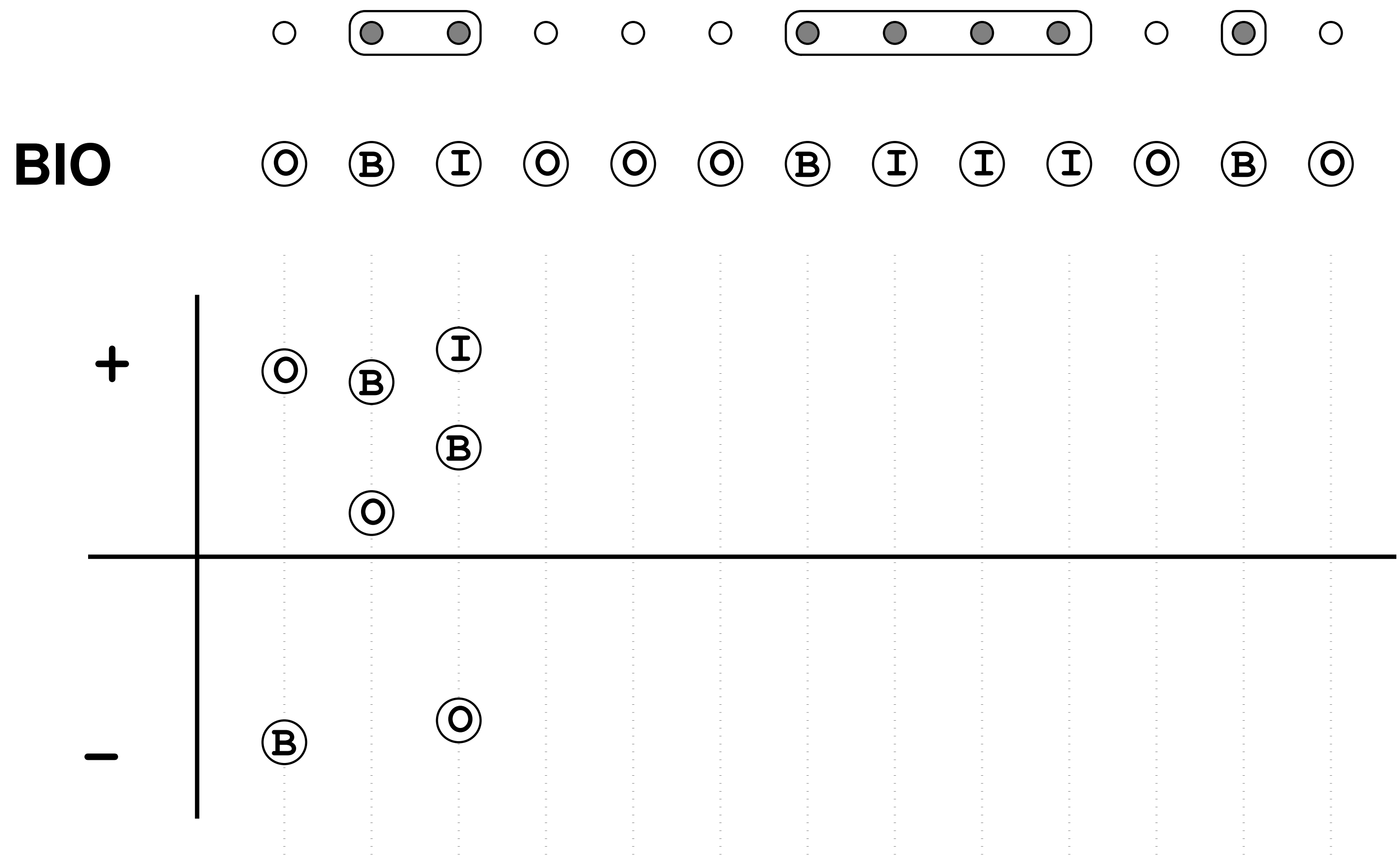
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



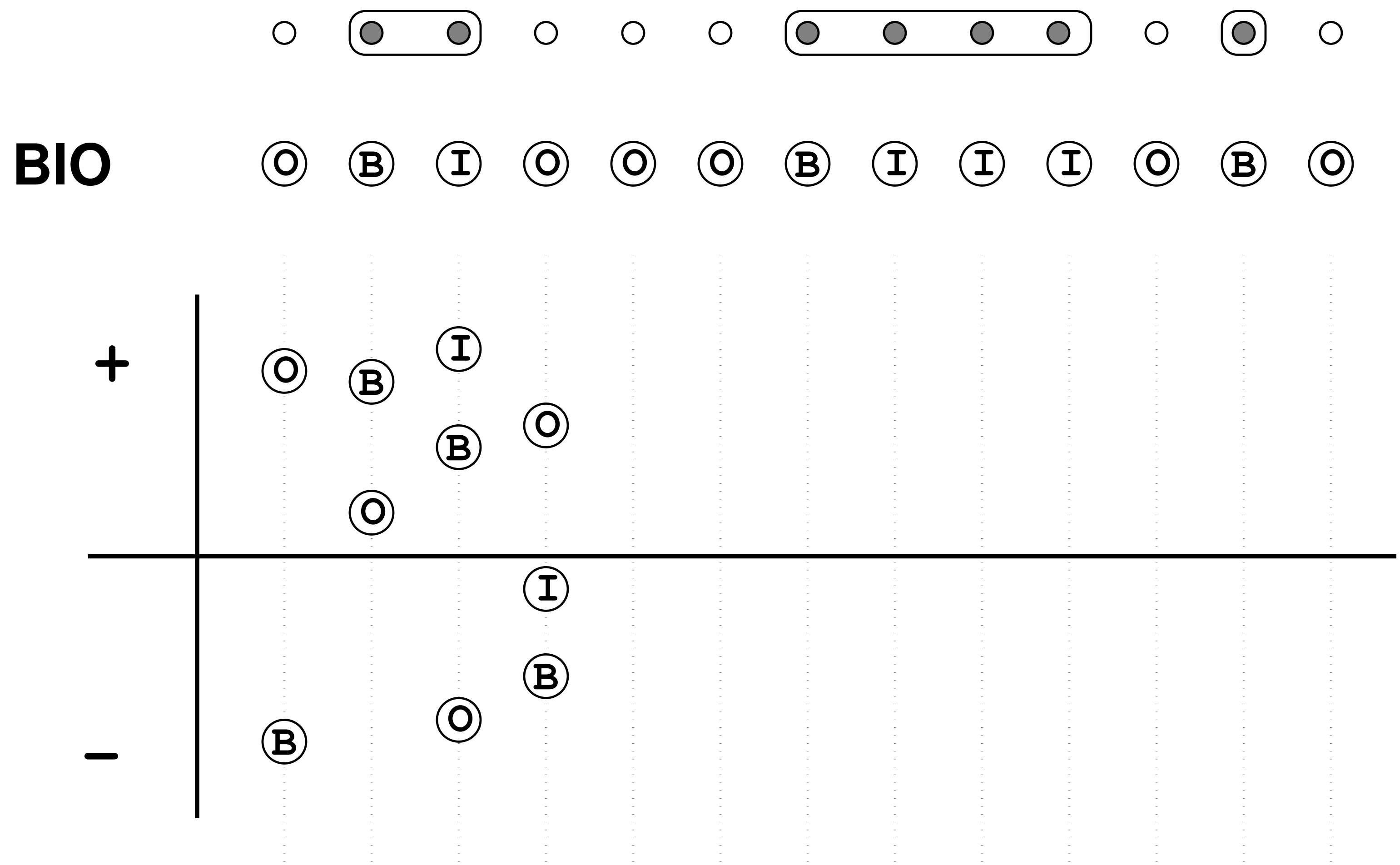
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



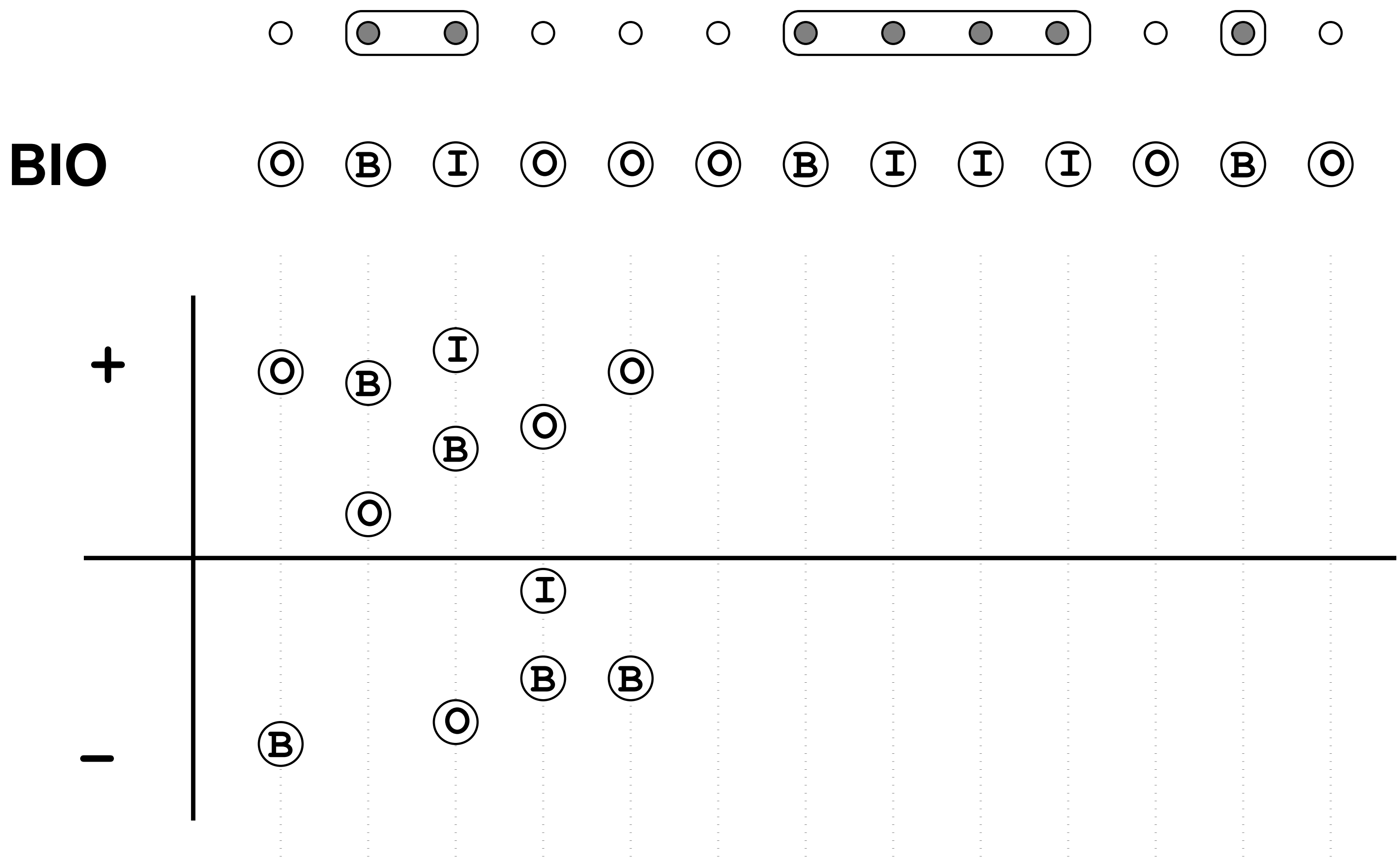
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



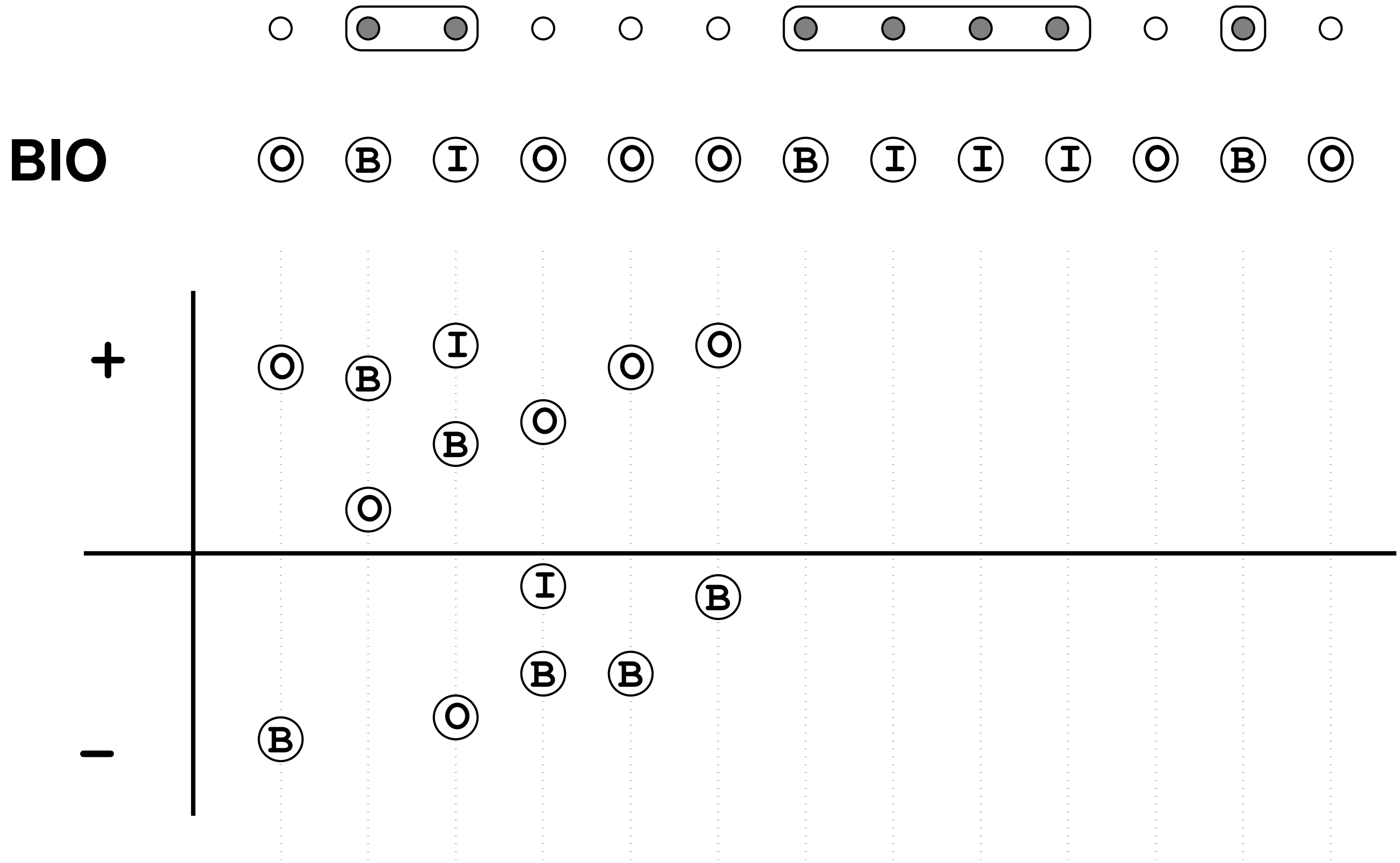
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



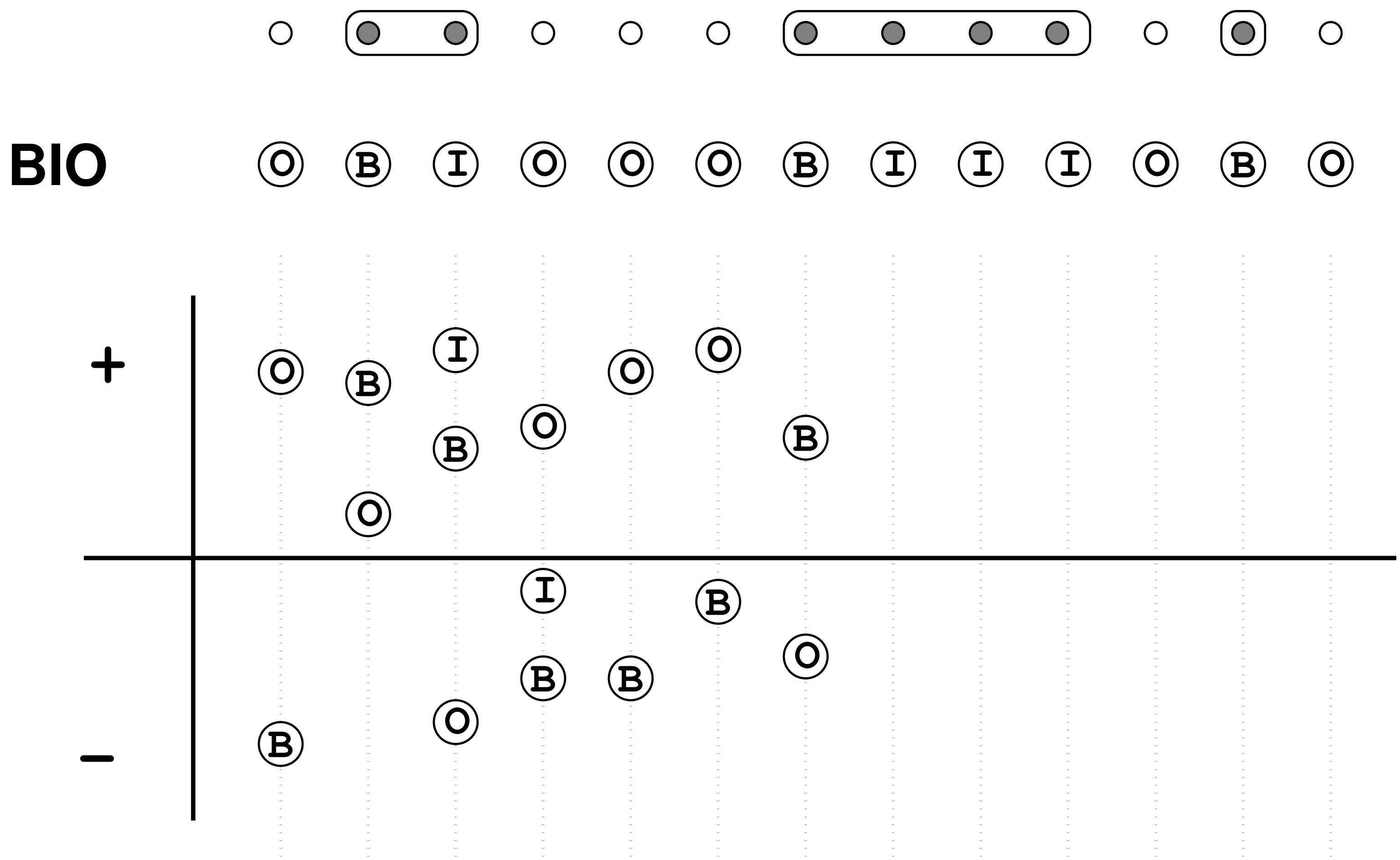
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



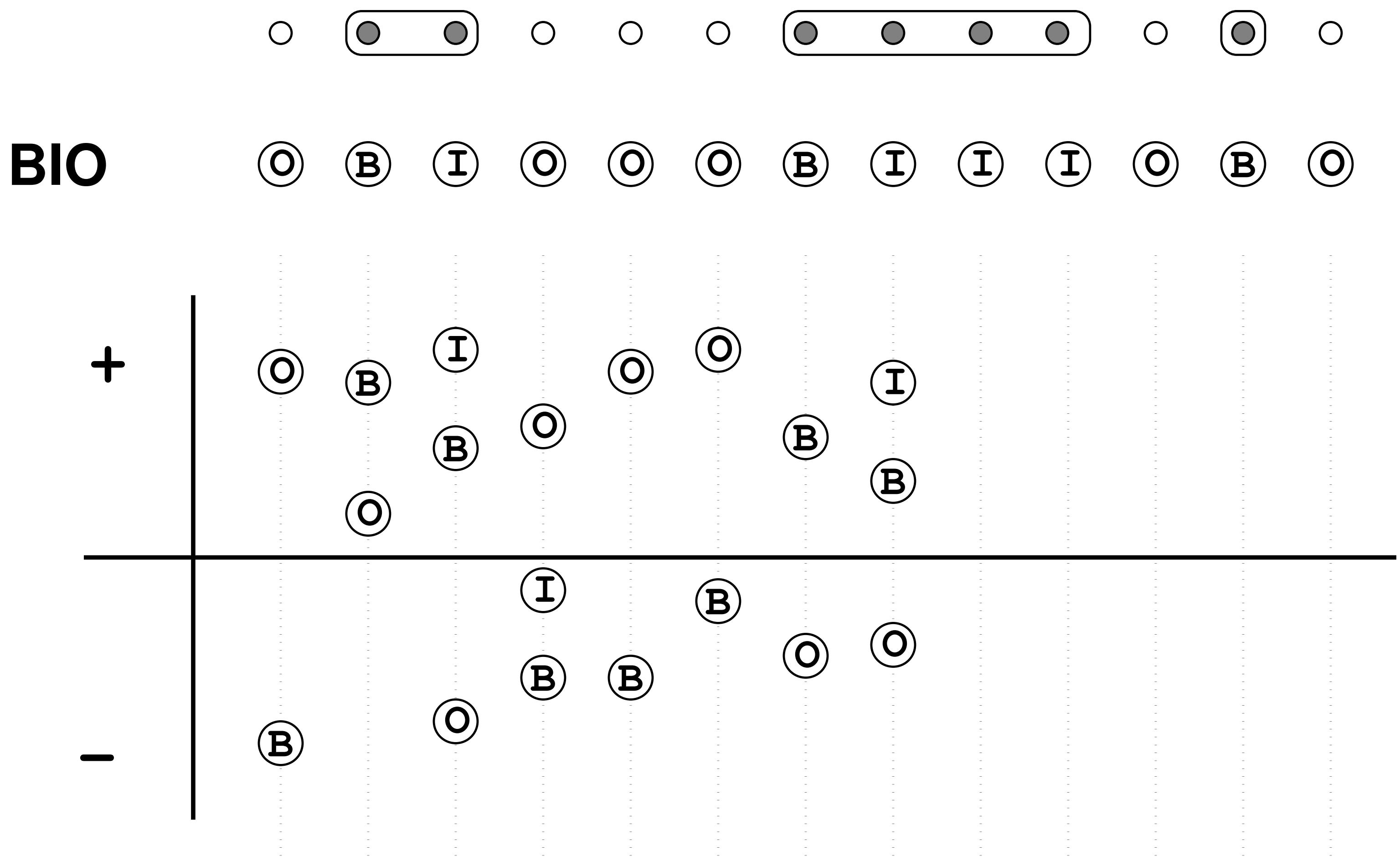
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



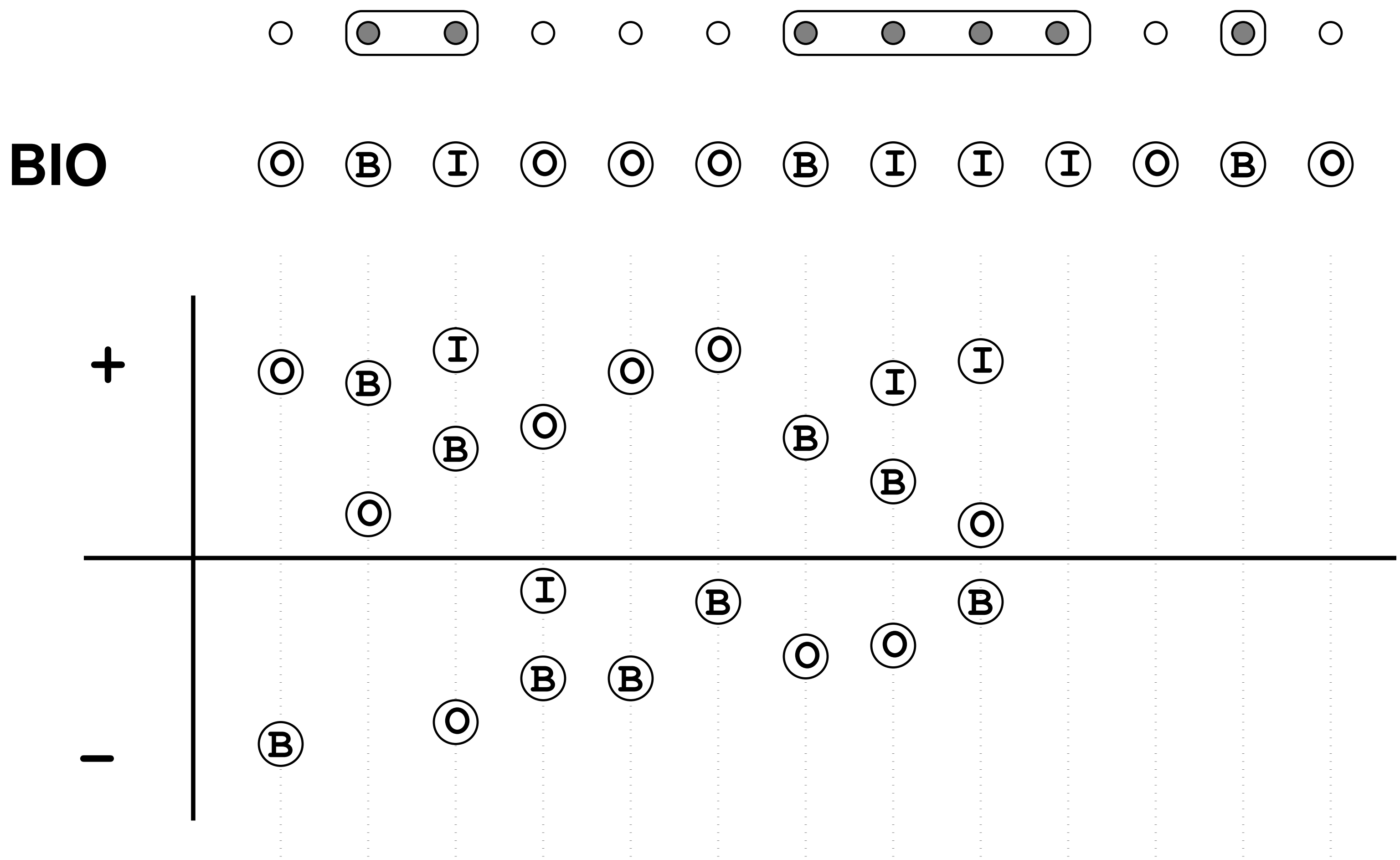
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



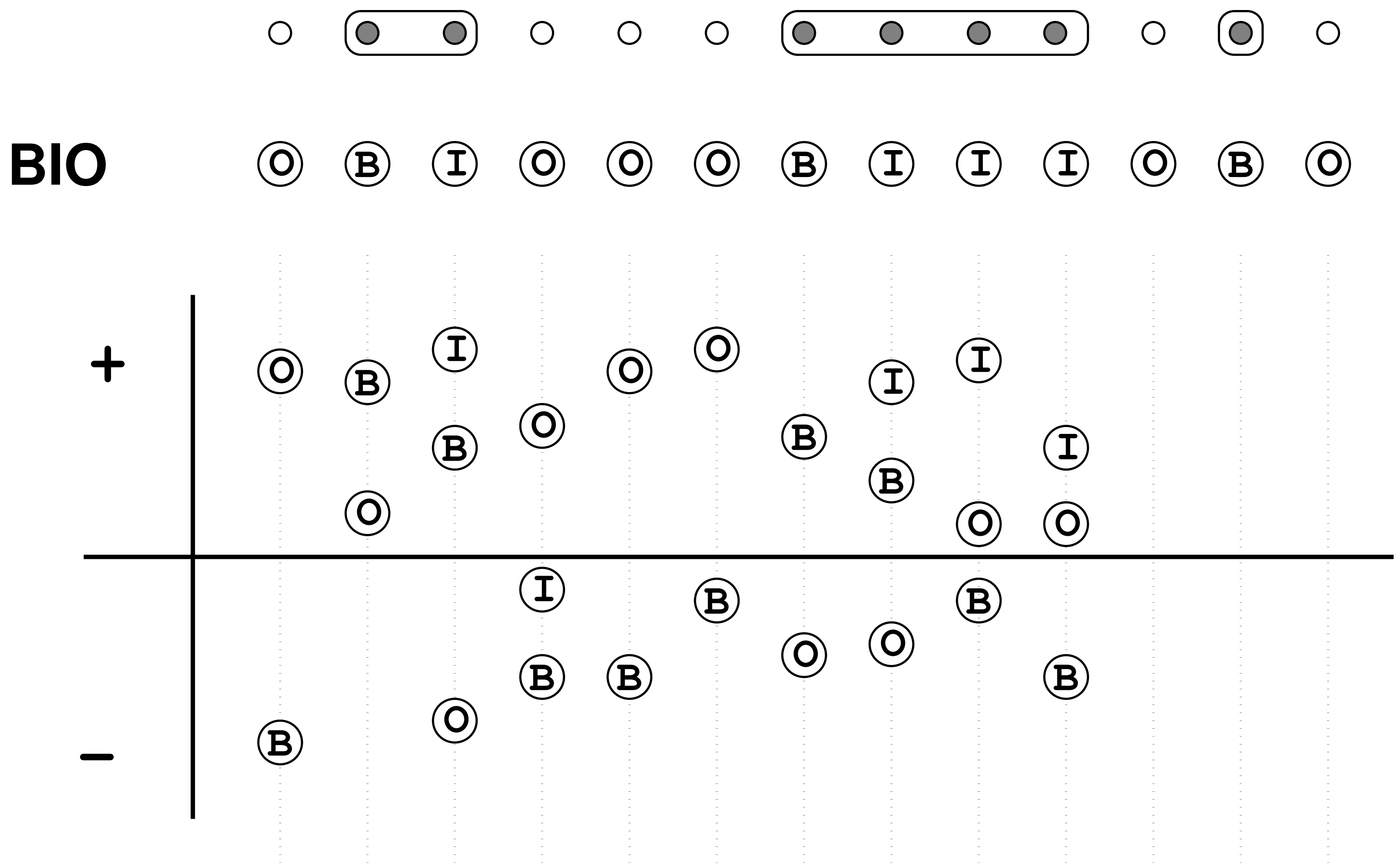
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



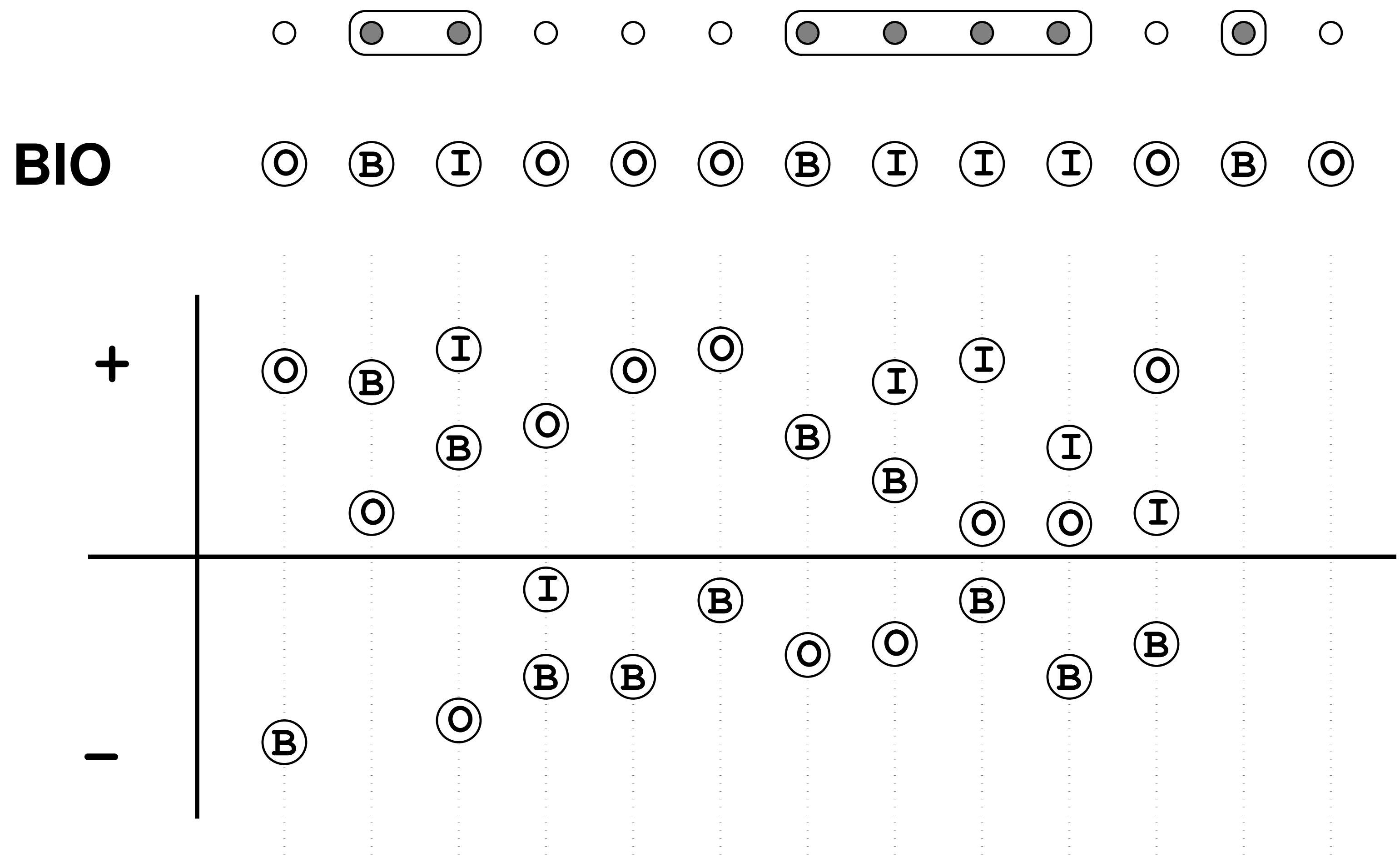
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



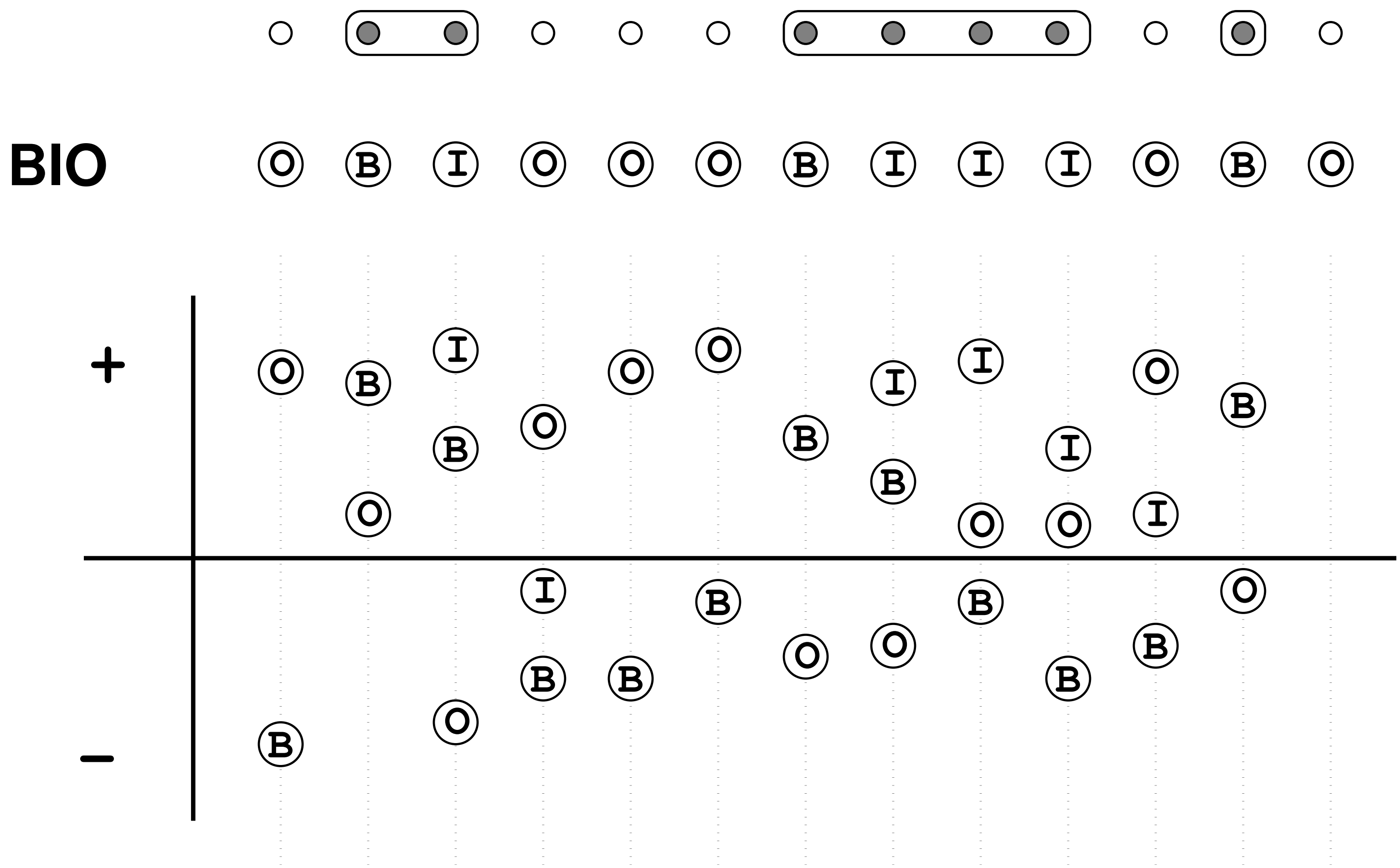
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



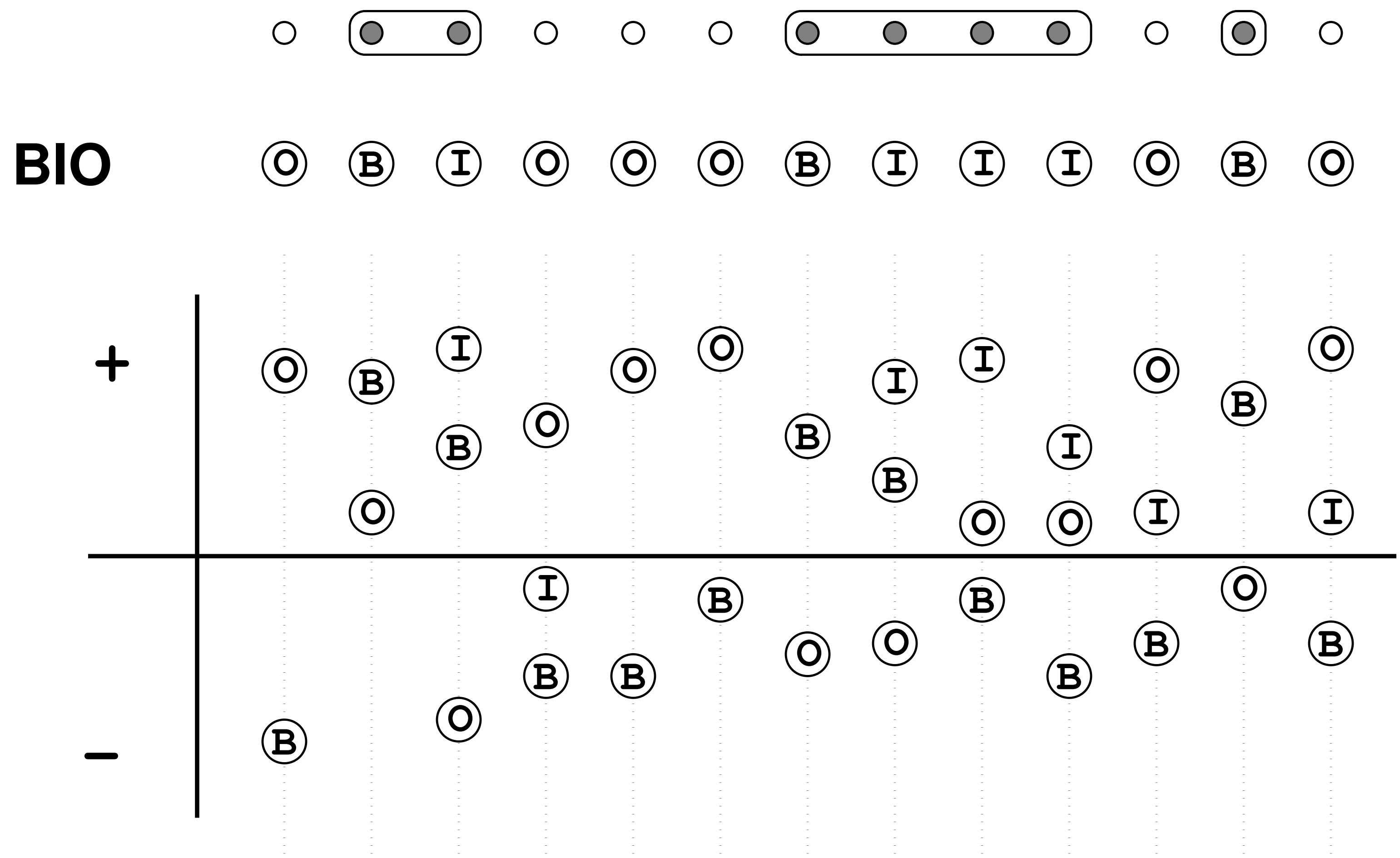
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



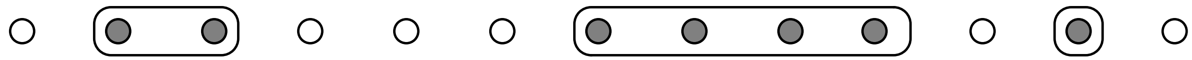
Learning and Inference: Simple Examples

BIO Tagging for Phrase Identification



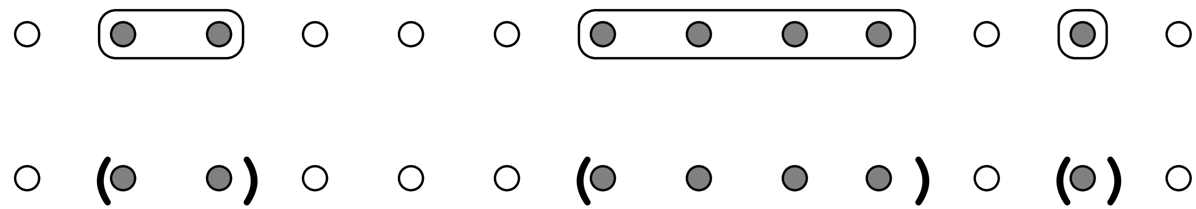
Learning and Inference: Simple Examples

Open-Close for Phrase Identification



Learning and Inference: Simple Examples

Open-Close for Phrase Identification

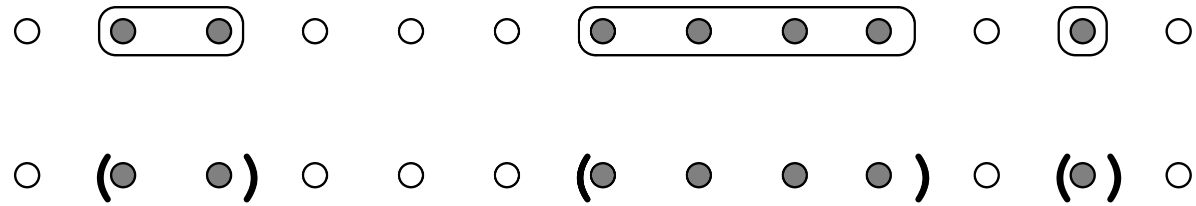


open

close

Learning and Inference: Simple Examples

Open-Close for Phrase Identification

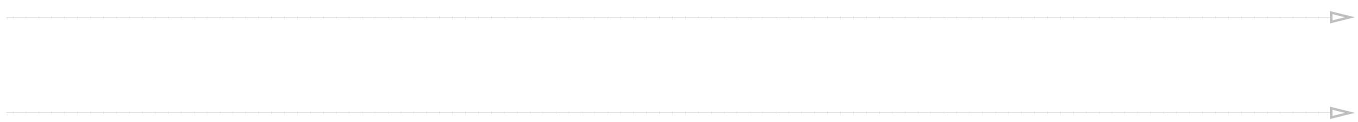


open

close

open

close

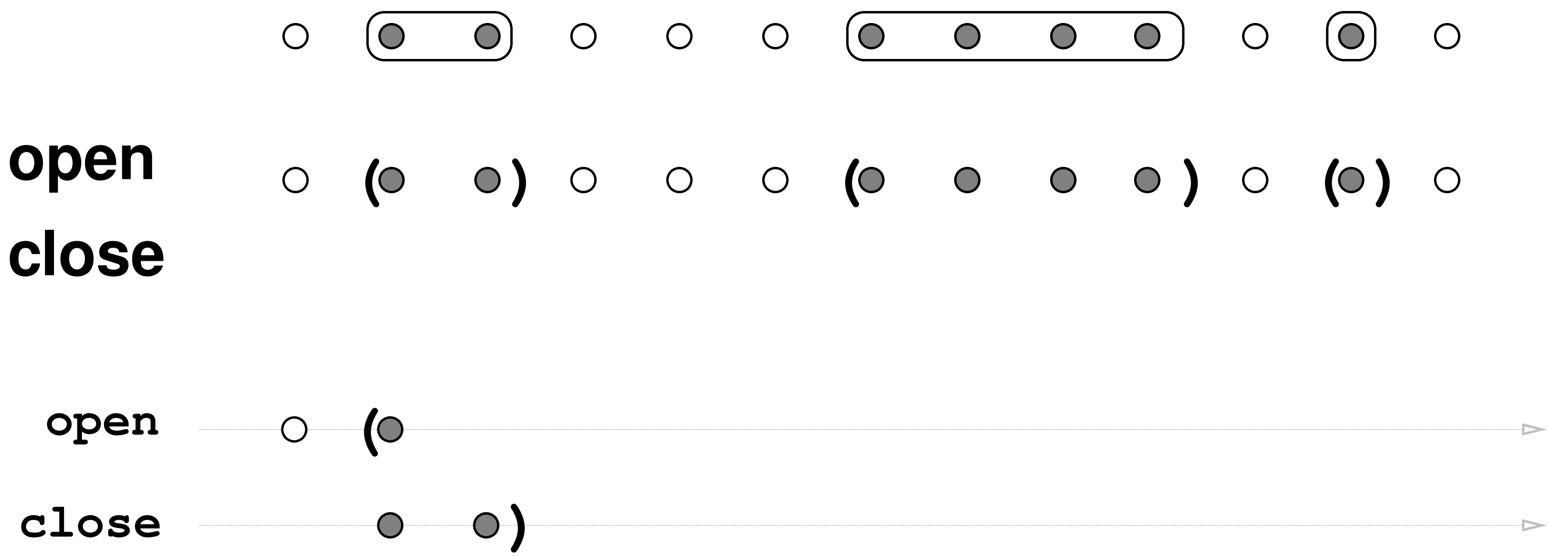


Learning and Inference: Simple Examples

Open-Close for Phrase Identification

Learning and Inference: Simple Examples

Open-Close for Phrase Identification

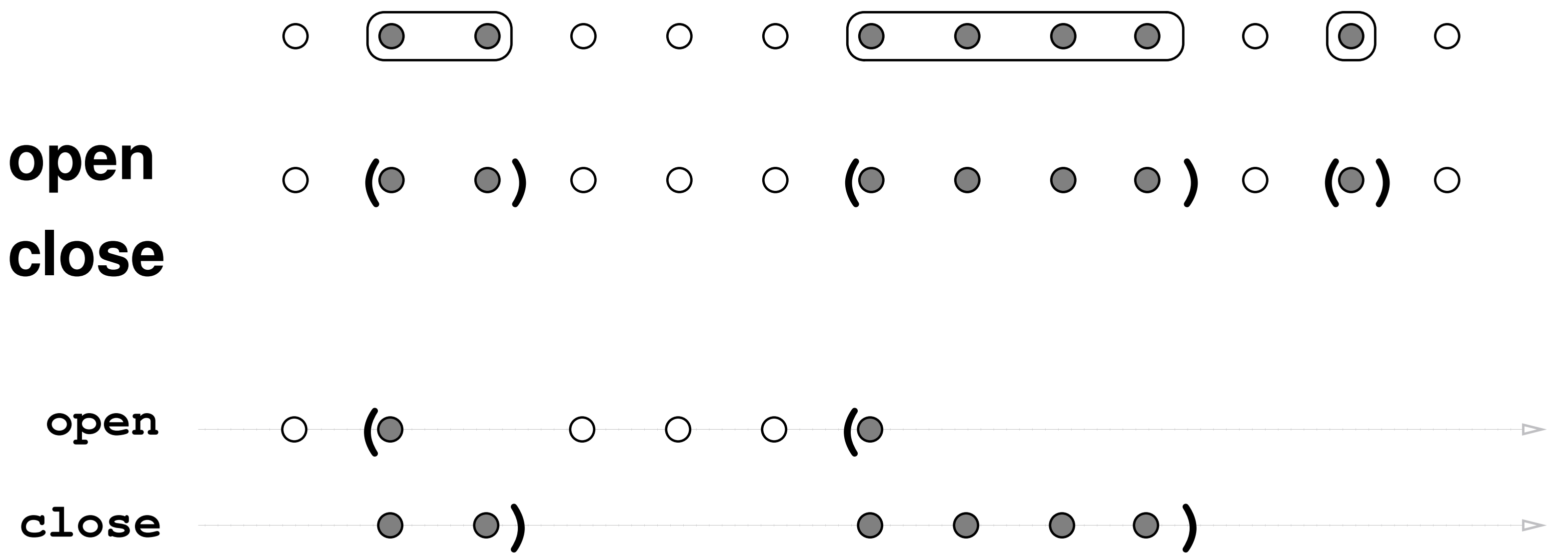


Learning and Inference: Simple Examples

Open-Close for Phrase Identification

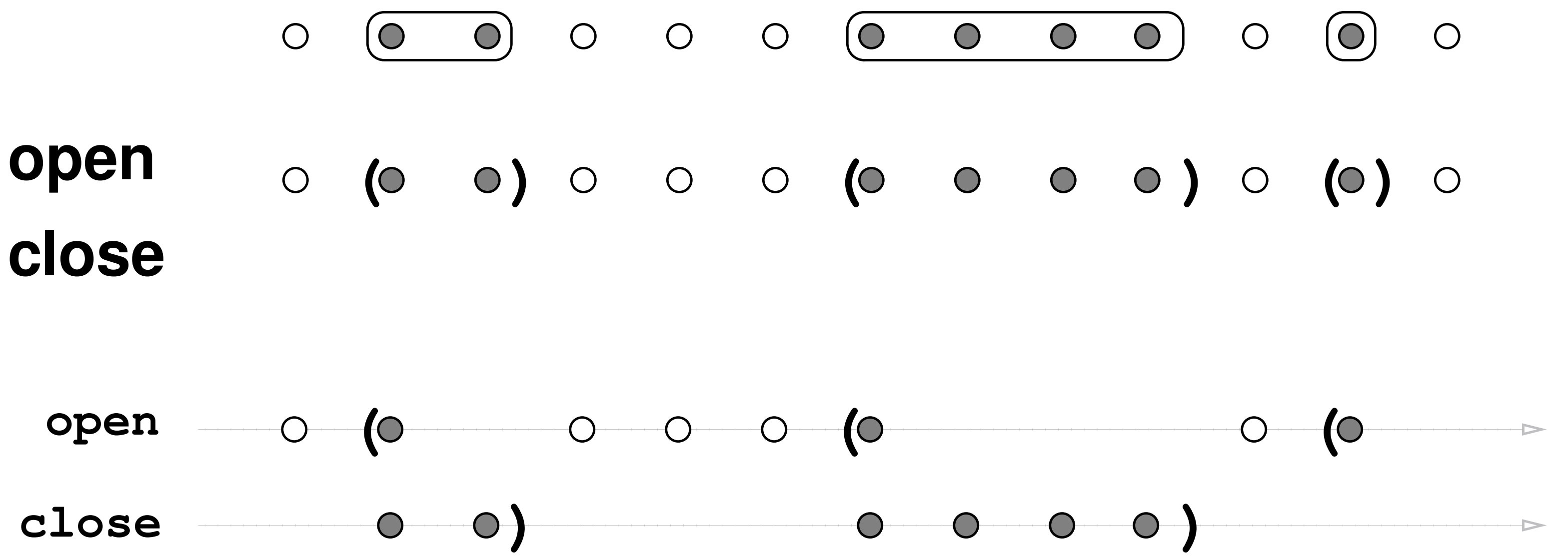
Learning and Inference: Simple Examples

Open-Close for Phrase Identification



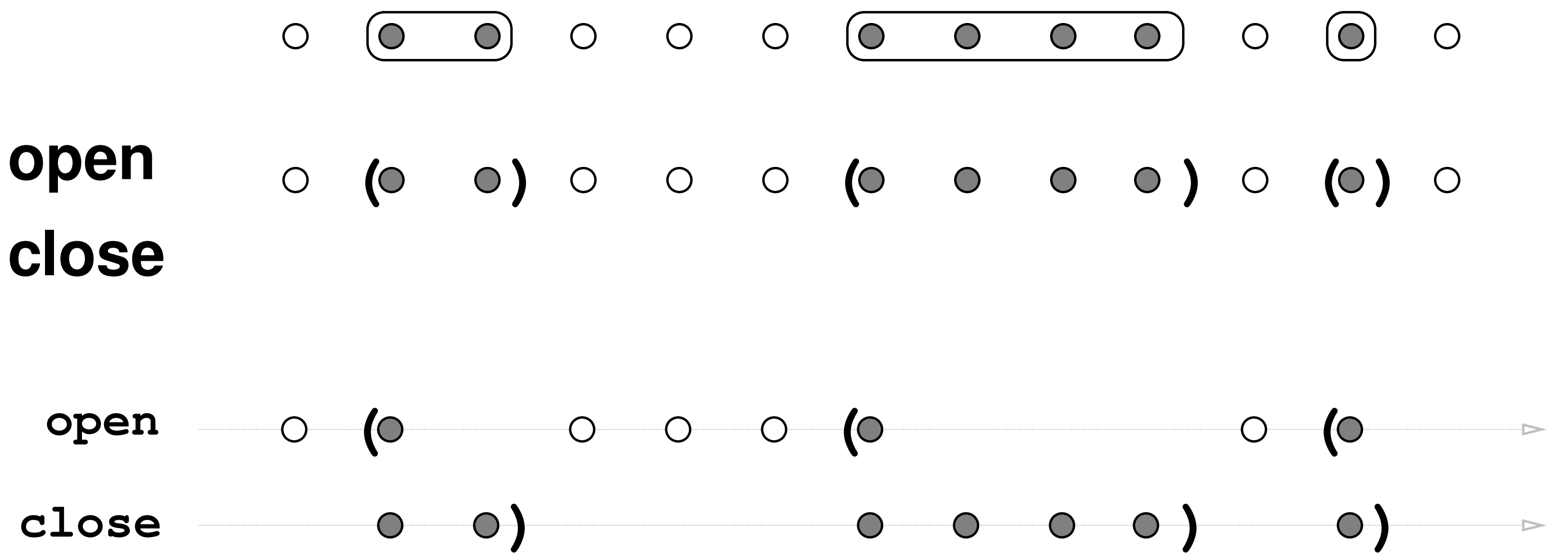
Learning and Inference: Simple Examples

Open-Close for Phrase Identification



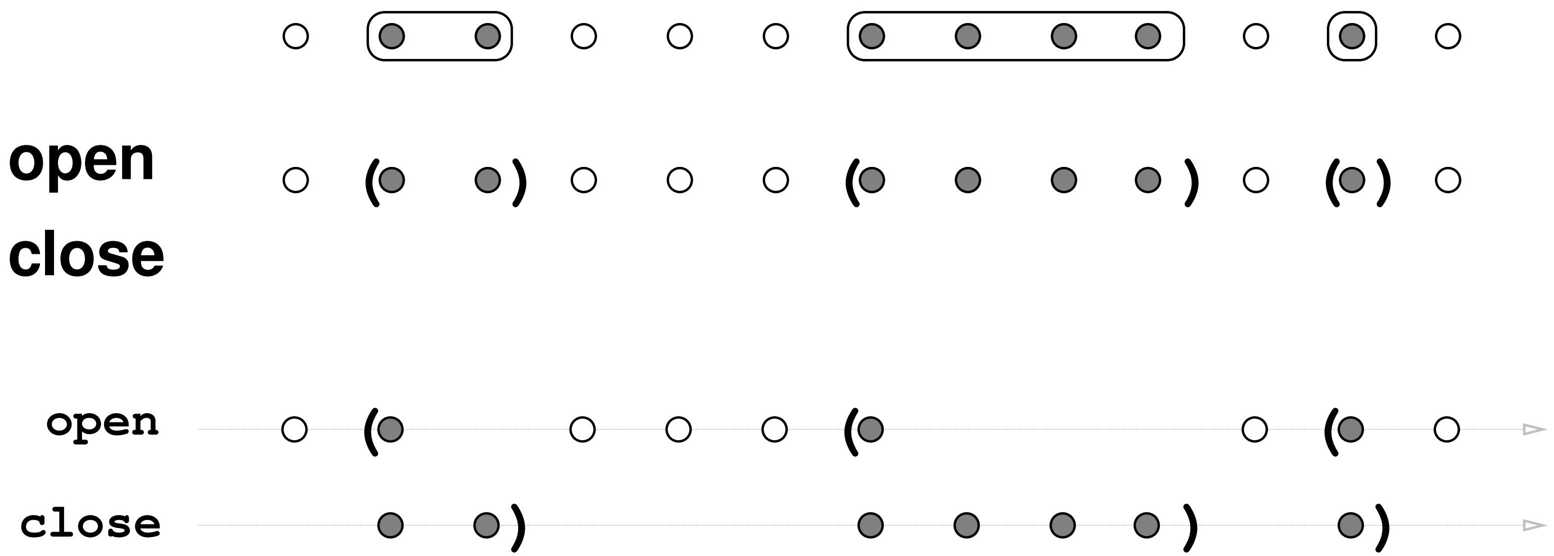
Learning and Inference: Simple Examples

Open-Close for Phrase Identification



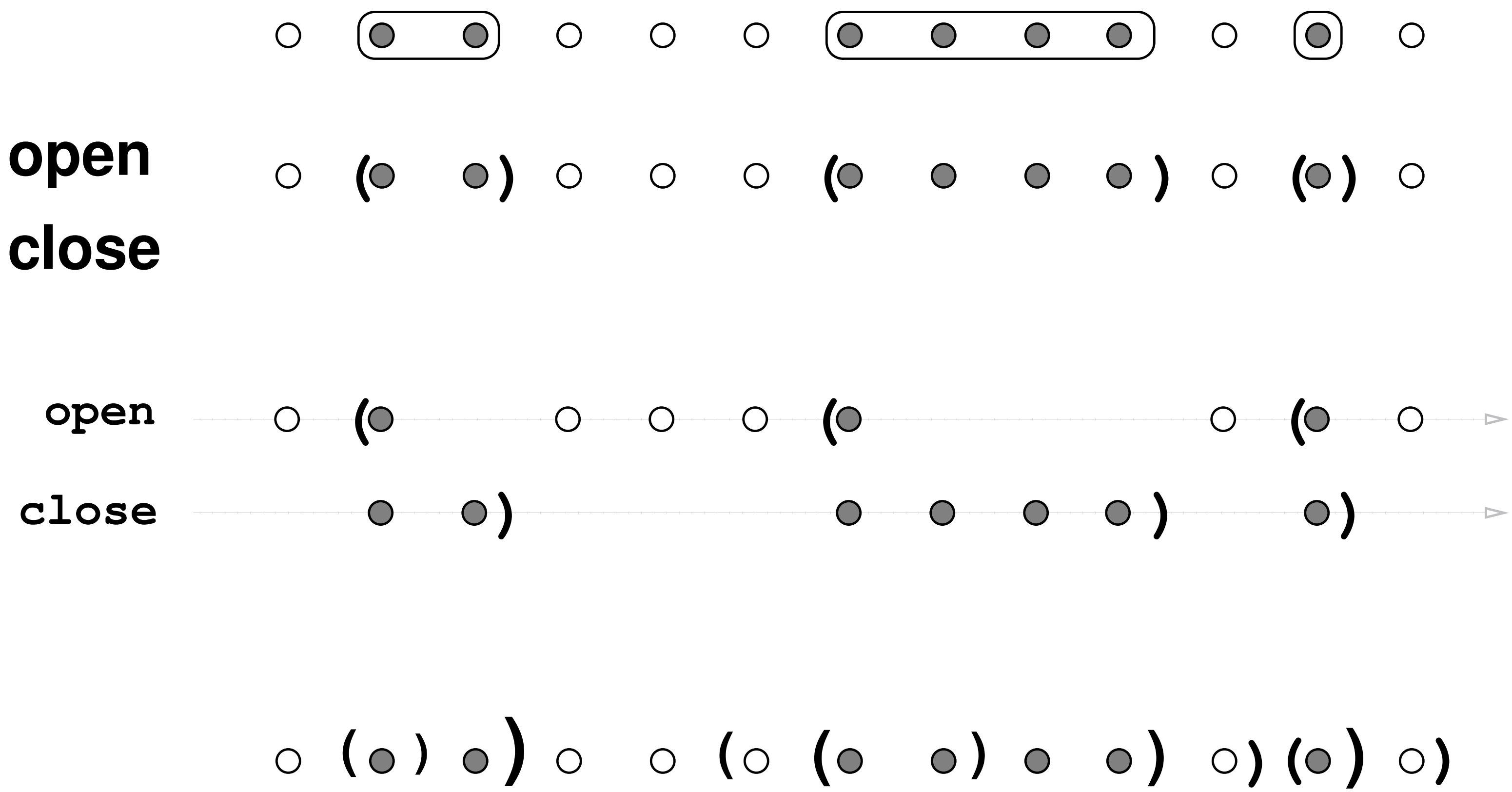
Learning and Inference: Simple Examples

Open-Close for Phrase Identification



Learning and Inference: Simple Examples

Open-Close for Phrase Identification



Learning and Inference (Roth et al.)

- **Divide and Conquer** strategy:
  - ⋆ Decomposition into a number of local decisions to learn (you can use any classifier that output confidence scores)
  - ⋆ Inference scheme to construct the solution on top of classifiers' predictions; possibly including constraints given by the problem **[Punyakanok and Roth, 2001; 2004; Yih and Roth, 2004]**

Sequential Phrase Identification

- Formalization and proposal of three decompositions and exact inference procedures **[Punyakanok & Roth, 2001; 2004]**
- **HMM with classifiers**:
  - ⋆ HMM: $P(y_1)$, $P(y_t|y_{t-1})$, $P(x_t|y_t)$
  - ⋆ $P(x_t|y_t) = \frac{P(y_t|x_t)P(x_t)}{P(y_t)}$
  - ⋆ Classifiers provide $P(y_t|x_t)$
  - ⋆ Actually, it is extended to $P(y_t|\hat{x}_t)$
  - ⋆ The objective function is exactly the same than in regular HMM's. Inference is done by using the Viterbi decoder

Sequential Phrase Identification

[Punyakanok & Roth, 2001; 2004]

- **Projective Markov Models (PMM)**:
  - ⋆ Classifiers directly estimate $P(y_t|y_{t-1}, \hat{x}_t)$
  - ⋆ Optionally, train:
    - ∗ a binary classifier for each pair (y_t, y_{t-1})
    - ∗ a binary classifier for each y including features on y_{t-1}
    - ∗ a single multiclass classifier including features on y_{t-1}
  - ⋆ Convert output scores in true probabilities (e.g., using $\mathrm{softmax}$)
  - ⋆ The objective functions is: $\arg\max_{y_1,\ldots,y_n} \prod_{k=1}^{n} P(y_t|y_{t-1}, \hat{x}_t)$
  - ⋆ The inference is again the Viterbi decoder

Sequential Phrase Identification

[Punyakanok & Roth, 2001; 2004]

- **Constraint Satisfaction with classifiers**:
  - ⋆ CSP problem casted as a DAG based on open-close
  - ⋆ Classifiers provide confidence on open and close decisions
  - ⋆ The inference is the *shortest path* algorithm
- **Empirical Results on the chunking task:**

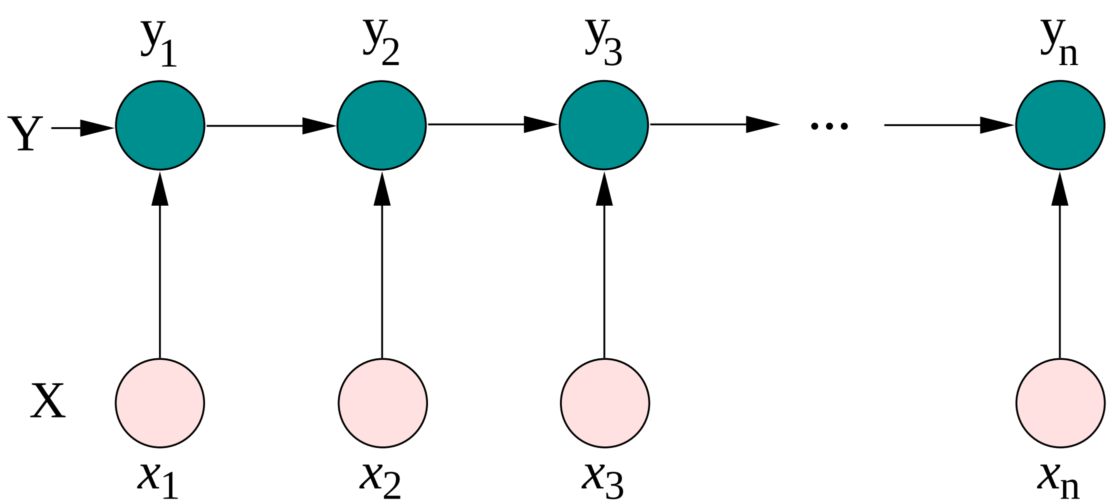
HMM $<$ HMM+class $<$ PMM $\approx$ CS+class

Sequential Phrase Identification

[Punyakanok & Roth, 2001; 2004]

- **Note$_1$**
  - ⋆ A PMM is also called Conditional Markov Model. Other examples using Maximum Entropy (MEMM) **[Ratnaparkhi 1996; 1999; McCallum et al.,2000]**

Graphical Model corresponding to a MEMM

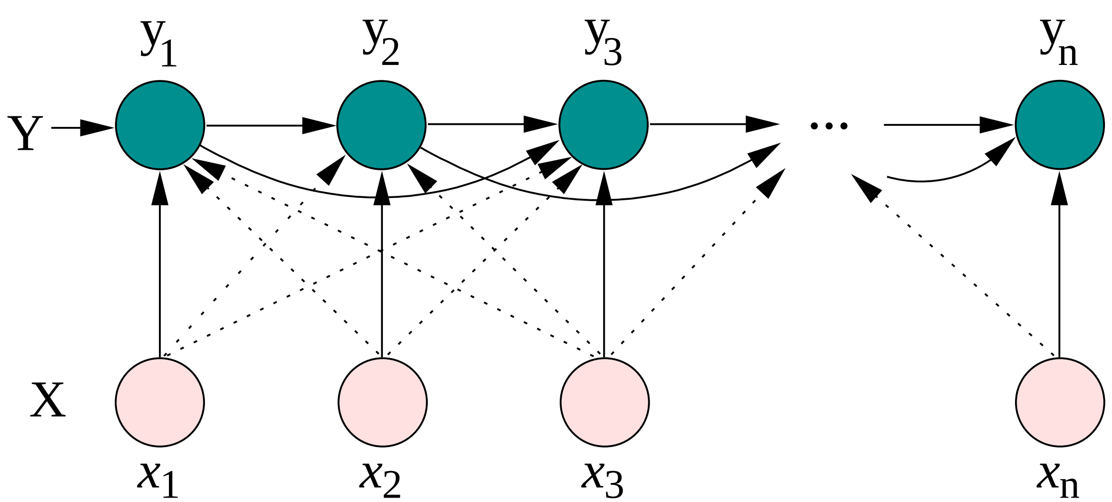


Sequential Phrase Identification

[Punyakanok & Roth, 2001; 2004]

- **Note$_1$**
  - ⋆ A PMM is also called Conditional Markov Model. Other examples using Maximum Entropy (MEMM) **[Ratnaparkhi 1996; 1999; McCallum et al.,2000]**

Graphical Model corresponding to a MEMM



Sequential Phrase Identification

[Punyakanok & Roth, 2001; 2004]

Extension of the previous work

- Work with general constraints, not only structural
  - ⋆ Joint recognition of Named Entities and Relations **[Yih and Roth, 2004]**. See slides on that paper (in PowerPoint)
  - ⋆ Application to Semantic Role Labeling **[Punyakanok et al., 2005]**. More on this in the last session
- Solved using Integer Linear Programming.
  Exact inference is feasible

Hierarchical Phrase Identification

Clause Identification

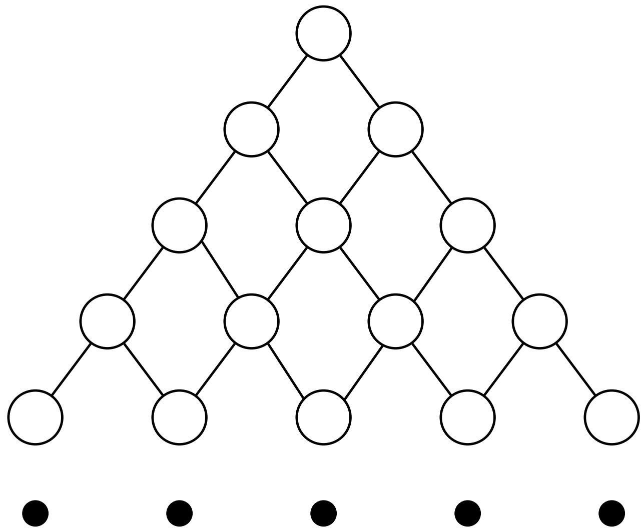
- Clause = sequence of contiguous words with a subject (maybe implicit) and a predicate.

((When (you don't have any other option)) ,
it's easy (to fight) .)

- A clause is represented by its boundary words (s, e).
- Clauses in a sentence form a hierarchical structure.
- Clauses can not overlap, but may be embedded.

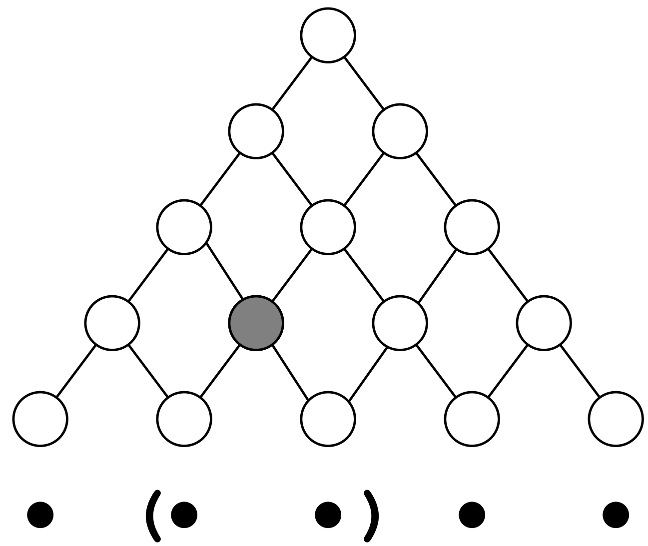
Hierarchical Phrase Identification

CI is a phrase recognition problem



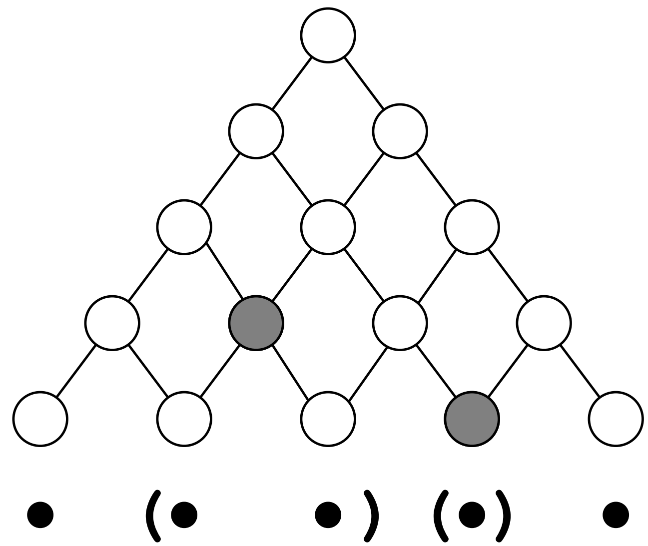
Hierarchical Phrase Identification

CI is a phrase recognition problem



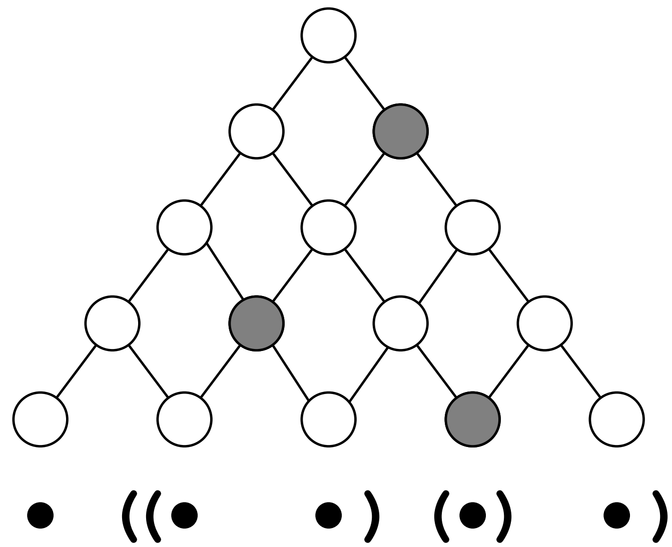
Hierarchical Phrase Identification

CI is a phrase recognition problem



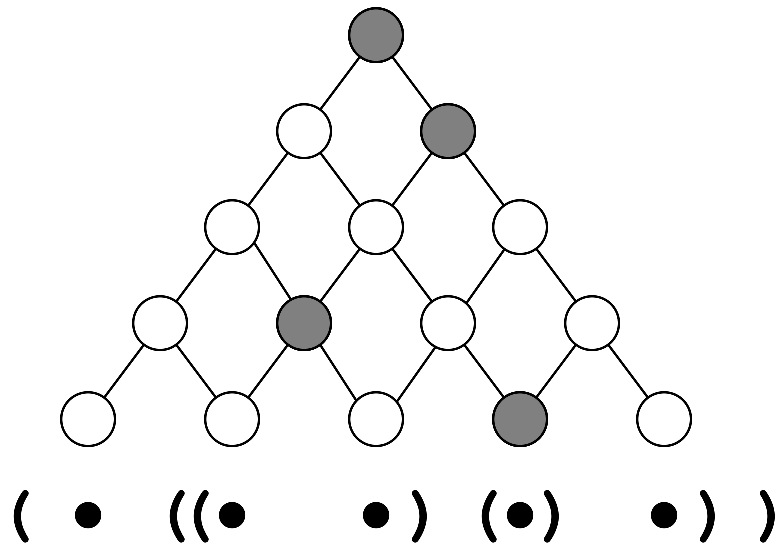
Hierarchical Phrase Identification

CI is a phrase recognition problem



Hierarchical Phrase Identification

CI is a phrase recognition problem



A solution is a coherent set of phrases

$$x = x_0, x_1, x_2, x_3, x_4$$
$$y = \{(3,3)_1, (1,2)_2, (1,4)_1, (0,4)_3\}$$

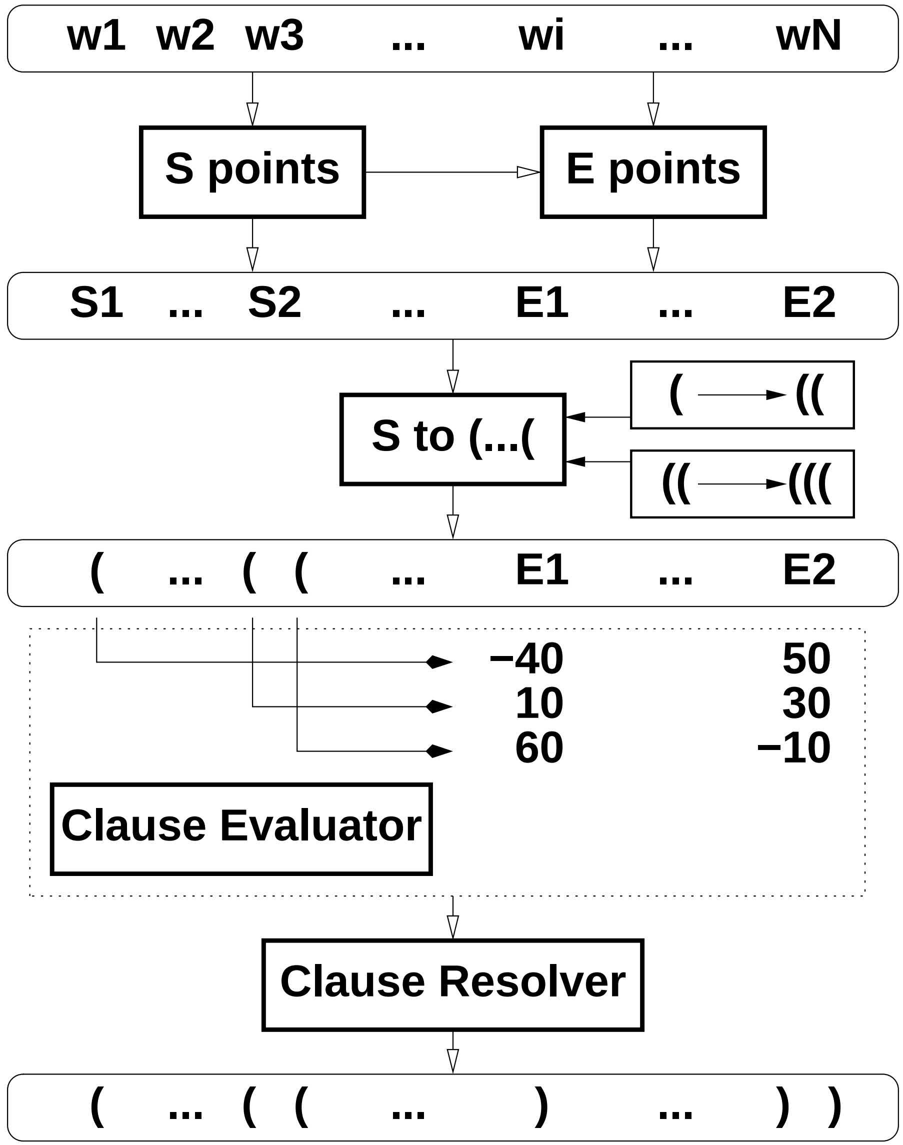
(a brief excursion) CoNLL-2001: Clause Splitting

- CoNLL series (1997–2006). Organized by ACL's SIGNLL (Special Interest Group on Natural Language Learning). http://www.cnts.ua.ac.be/conll/

- $\sim$2 months for developing and testing the system!

- **[Carreras and Màrquez, 2001]**:
  - ⋆ Decomposition of the whole problem into a combination of "simple" binary decisions
  - ⋆ AdaBoost algorithm (Shapire and Singer, 1999): Decision Trees of fixed depth (1-4)

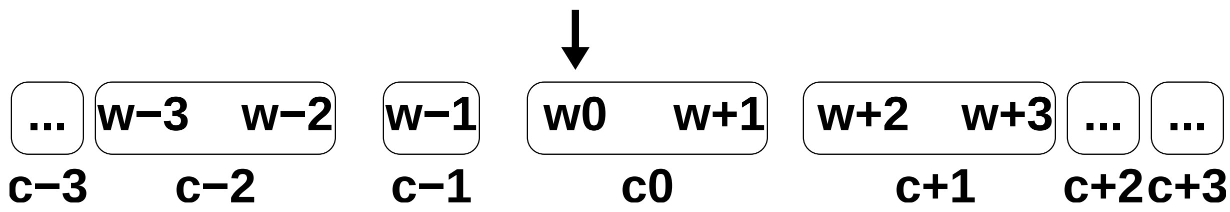
Hierarchical Phrase Identification: CoNLL-2001 Prototype



(a brief excursion) CoNLL-2001: Clause Splitting

Features

- Window:
  - ⋆ Word Window: word forms and POS.
  - ⋆ Chunk Window: chunk labels.



- Whole Sentence:
  - ⋆ Sentence patterns, representing the relevant structure of the sentence.

(a brief excursion) CoNLL-2001: Clause Splitting

Features

- Whole Sentence:
  - ⋆ Sentence patterns, representing the relevant structure of the sentence. Example:
    `The deregulation of railroads` **and** `trucking companies` **that began** `in 1980` **enabled** `shippers to` **bargain** `for transportation .`
    Pattern: `and that VP VP VP .`
- Sentence features (global):
  - ⋆ Verbs, Punctuation marks
  - ⋆ Relative pronouns, S points, and E points

(a brief excursion) CoNLL-2001: Clause Splitting

Sizes and Results

	#ex	#ex +	#feat	#wr	d
S points	138,069	22,950	43,356	2,000	3
E points	138,069	17,150	42,853	2,000	3
($\rightarrow$ ((	23,075	1,519	36,240	1,500	3
(($\rightarrow$ (((	1,600	81	5,361	100	3
Cl.Eval.	39,209	16,294	14,040	2,000	3

development	precision	recall	$F_{\beta=1}$
part 1	95.77%	92.08%	93.89%
part 2	91.27%	89.00%	90.12%
part 3	87.18%	82.48%	**84.77%**

(a brief excursion) CoNLL-2001: Clause Splitting

Conclusions

- The system did not perform bad at all...
  - ⋆ second best system performed about 10 points lower!
- Why?
  - ⋆ Rich feature set; **Working at clause structure level**; A robust learning algorithm
- But...
  - ⋆ The system is ad–hoc and follows a greedy approach; Non–recursive decomposition of the task; The pipeline of decisions propagates errors

Filtering-Ranking Architecture

[Carreras et al., 2005]

- A general architecture to recognize phrase structures
- Two levels of learning:
  - ⋆ Filter: decides which words start/end a phrase
  - ⋆ Ranker: scores phrases
- On the top, dynamic programming inference builds the best-scored phrase structure
- We propose FR-Perceptron: a Perceptron learning algorithm tailored for the architecture

Filtering-Ranking Architecture: Decomposition

- A solution is decomposed at phrase level:

$$\mathrm{score}(x, y) = \sum_{(s,e)_k \in y} \mathrm{score_p}(x, y, (s, e)_k)$$

- Still, the number of phrases grows quadratically with the sentence length

- We reduce the space of phrases by filtering at word level.
  For a phrase $(s, e)_k$ to be in a solution:

$$\mathrm{start_w}(x, s, k) > 0 \quad \wedge \quad \mathrm{end_w}(x, e, k) > 0$$

Filtering-Ranking Model

$\mathcal{Y}$: solution space, i.e. set of all phrase structures

$\mathcal{Y}_{\mathrm{SE}}$: practical solution space, filtered at word level:

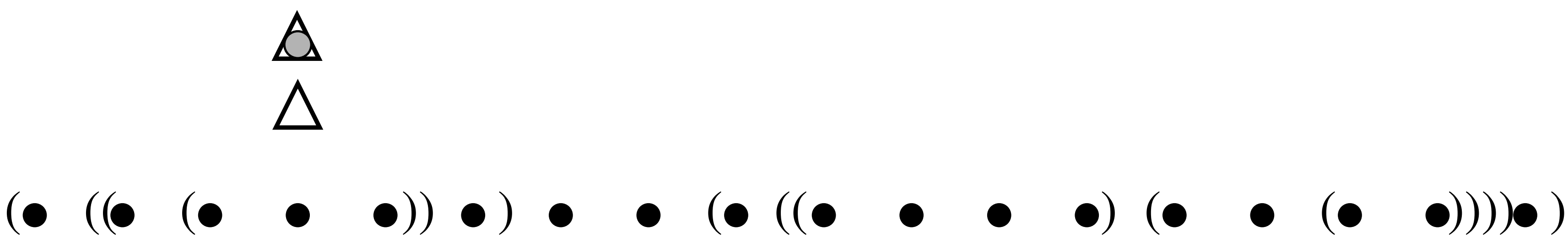
$$\mathcal{Y}_{\mathrm{SE}} = \{y \in \mathcal{Y} \mid \forall (s,e)_k \in y \;\; \mathrm{start_w}(x,s,k) \wedge \mathrm{end_w}(x,e,k)\}$$

The Filtering-Ranking architecture computes:

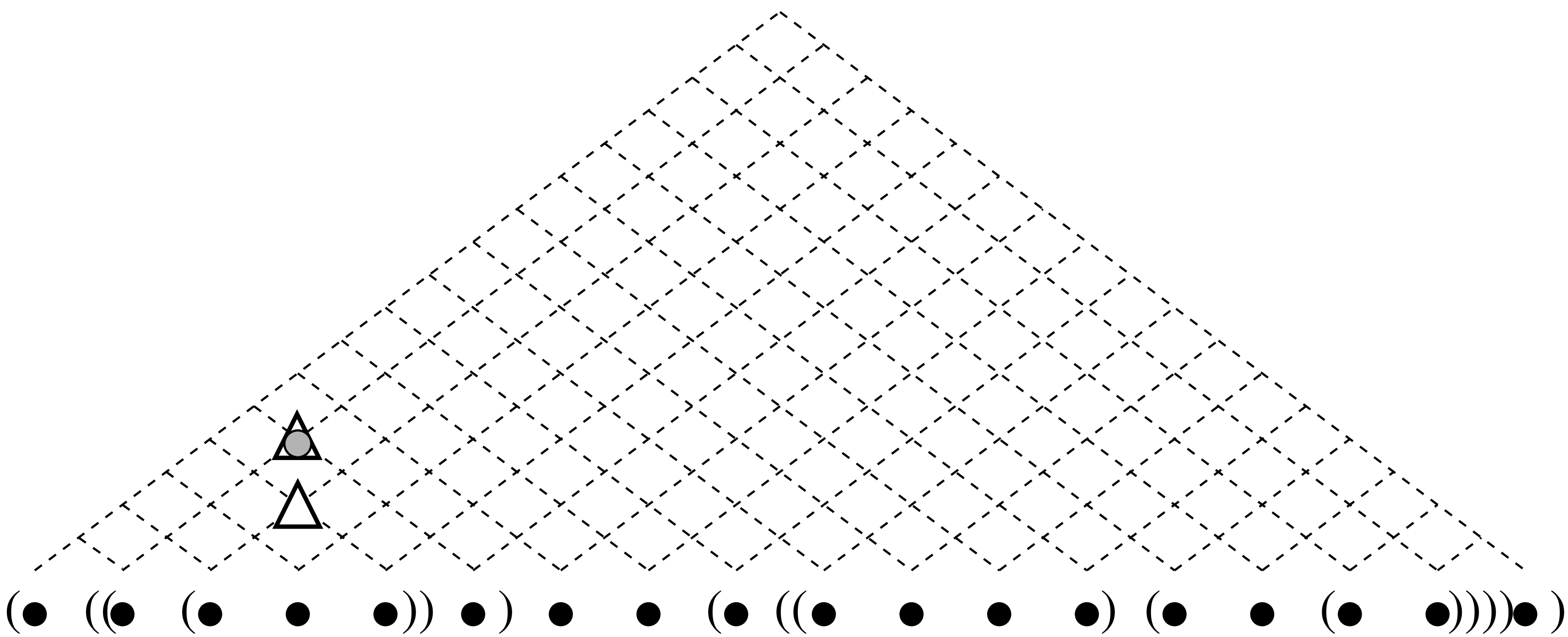
$$\mathrm{PhRec}(x) = \arg\max_{y \in \mathcal{Y}_{\mathrm{SE}}} \sum_{(s,e)_k \in y} \mathrm{score_p}(x, y, (s,e)_k)$$

using dynamic-programming.

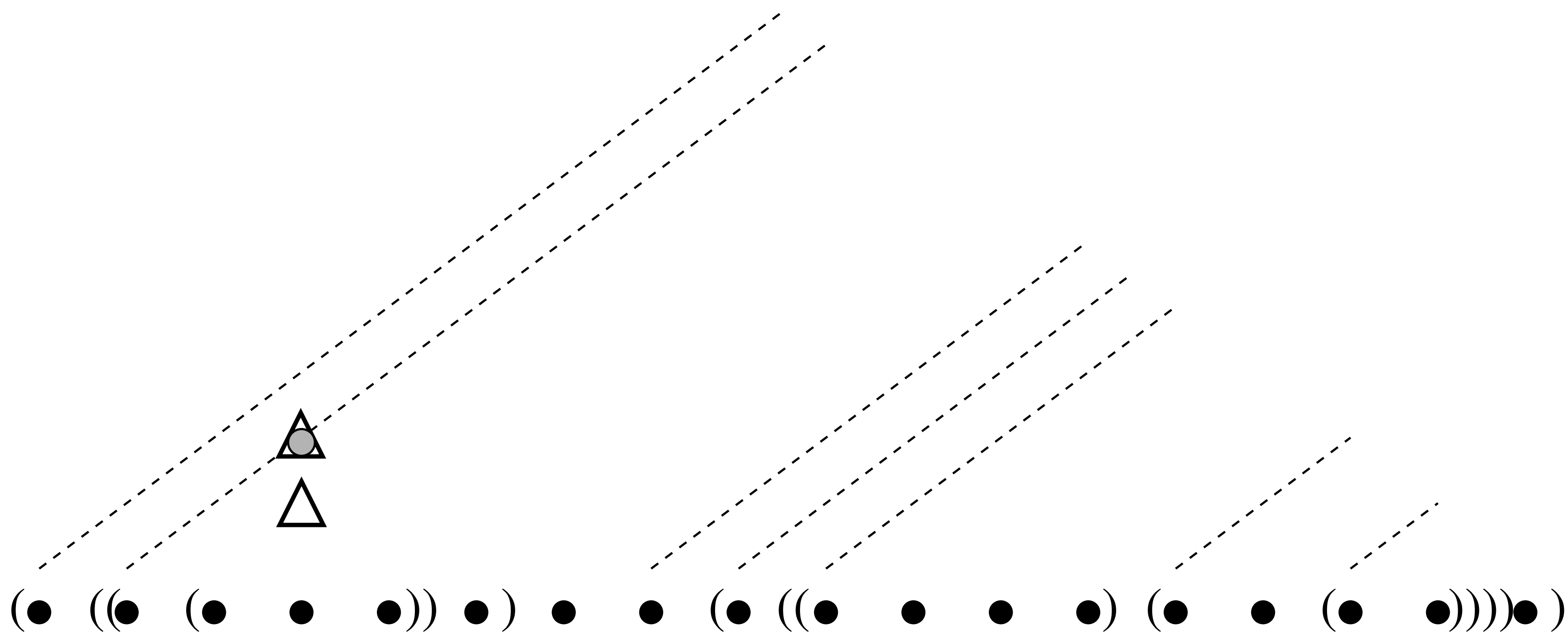
Filtering-Ranking Strategy



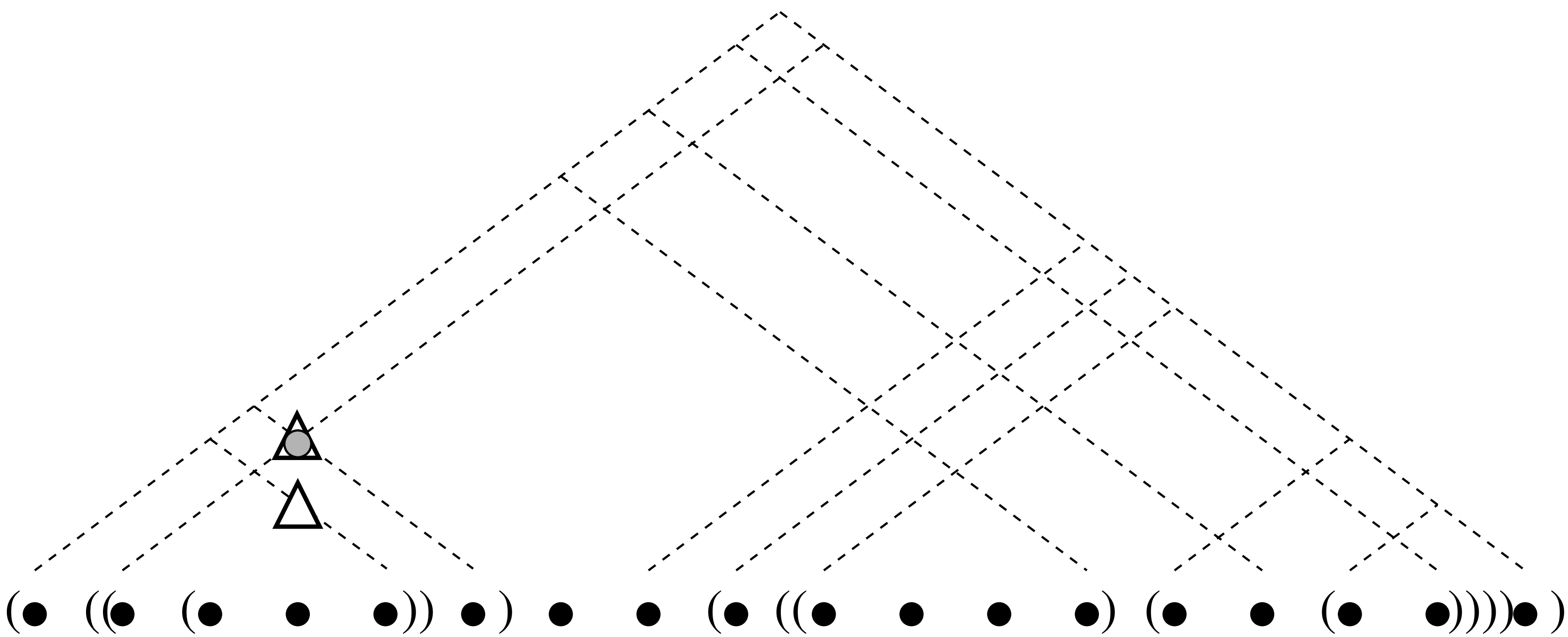
Filtering-Ranking Strategy



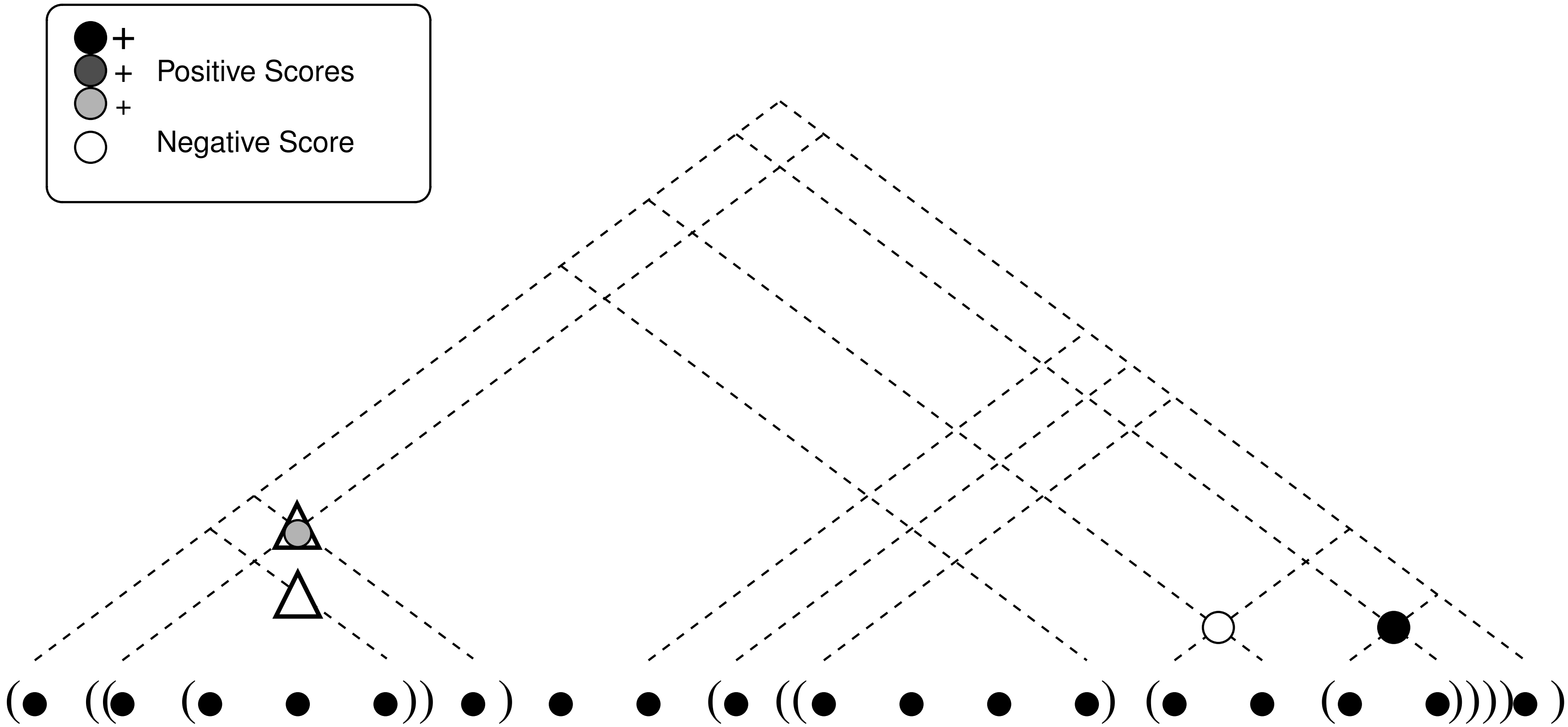
Filtering-Ranking Strategy



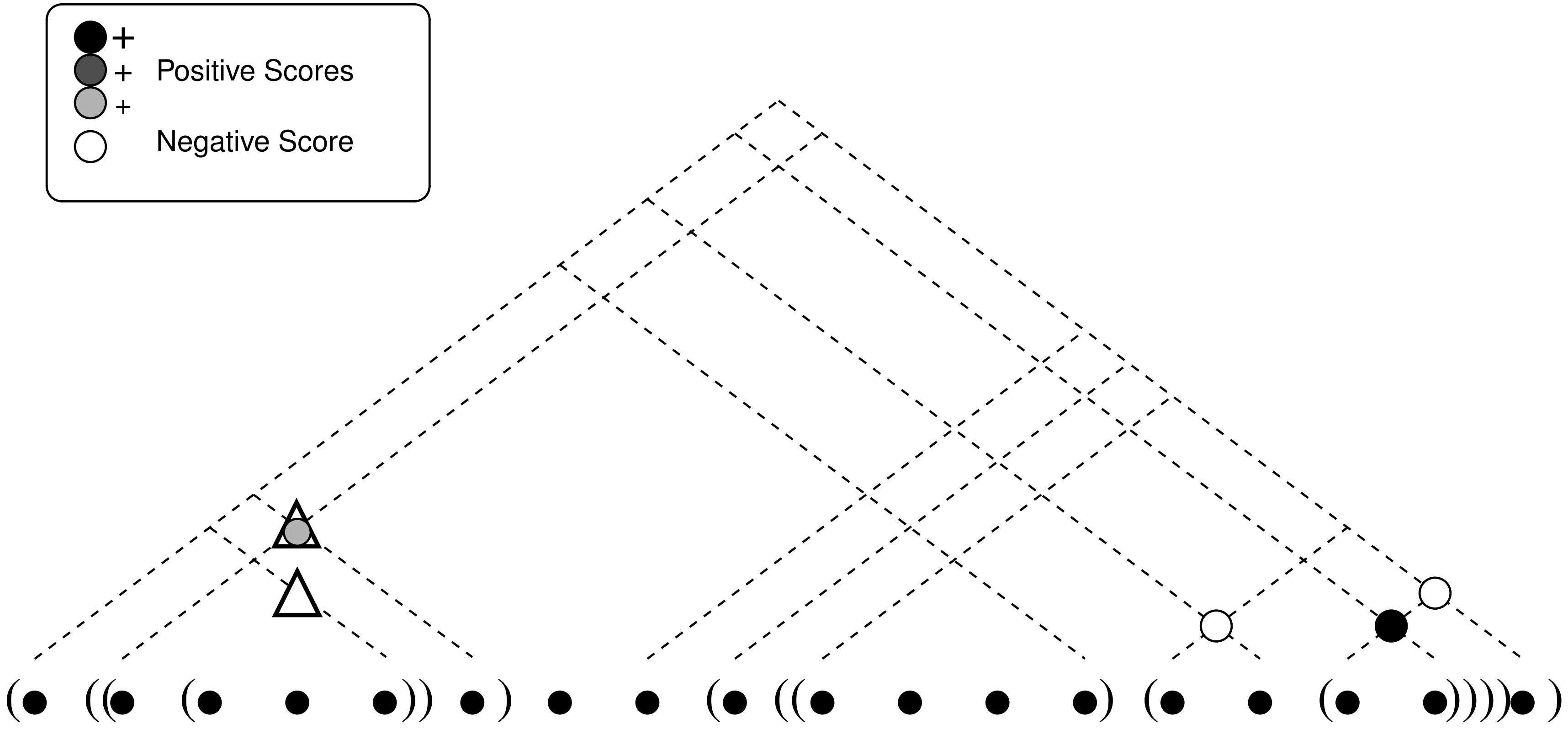
Filtering-Ranking Strategy



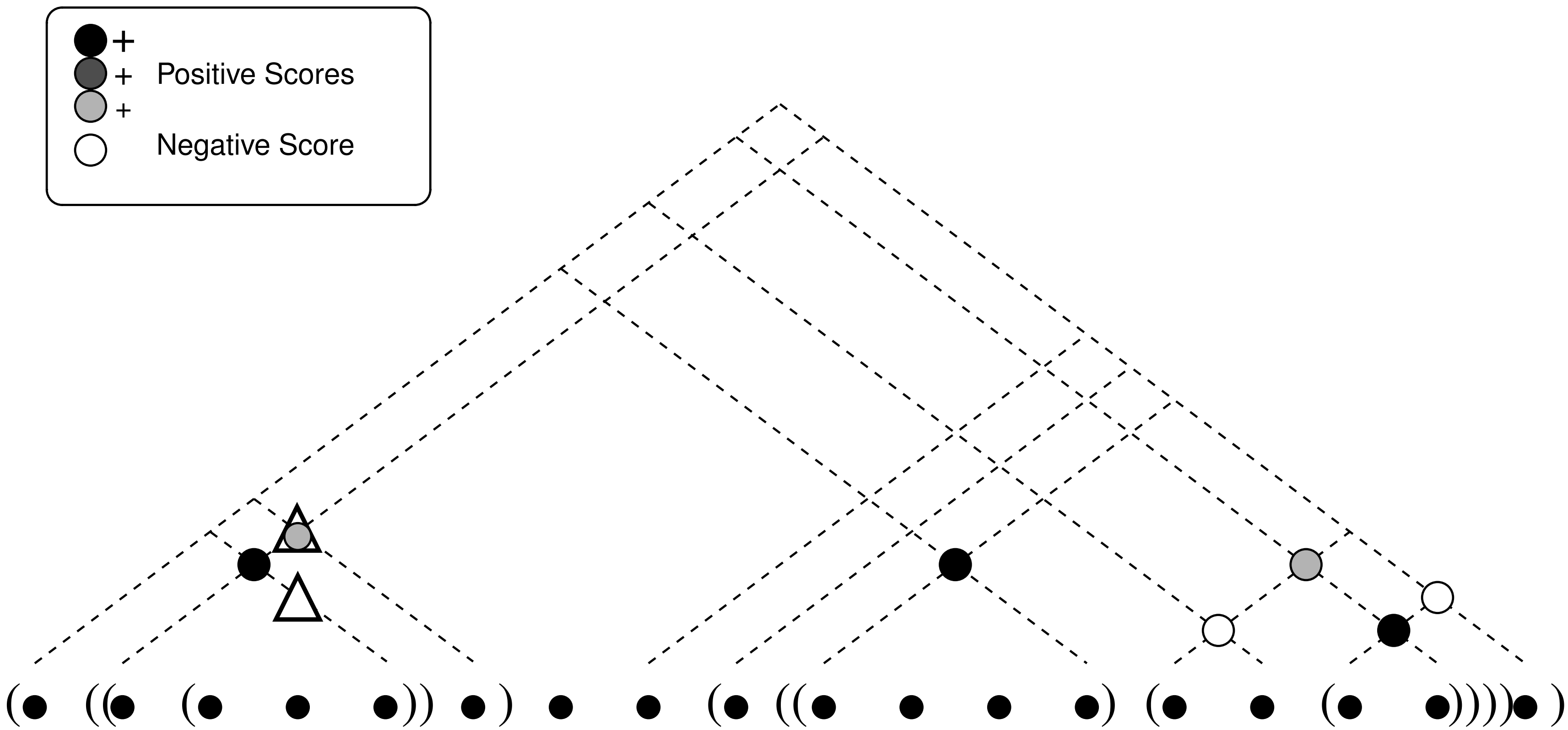
Filtering-Ranking Strategy



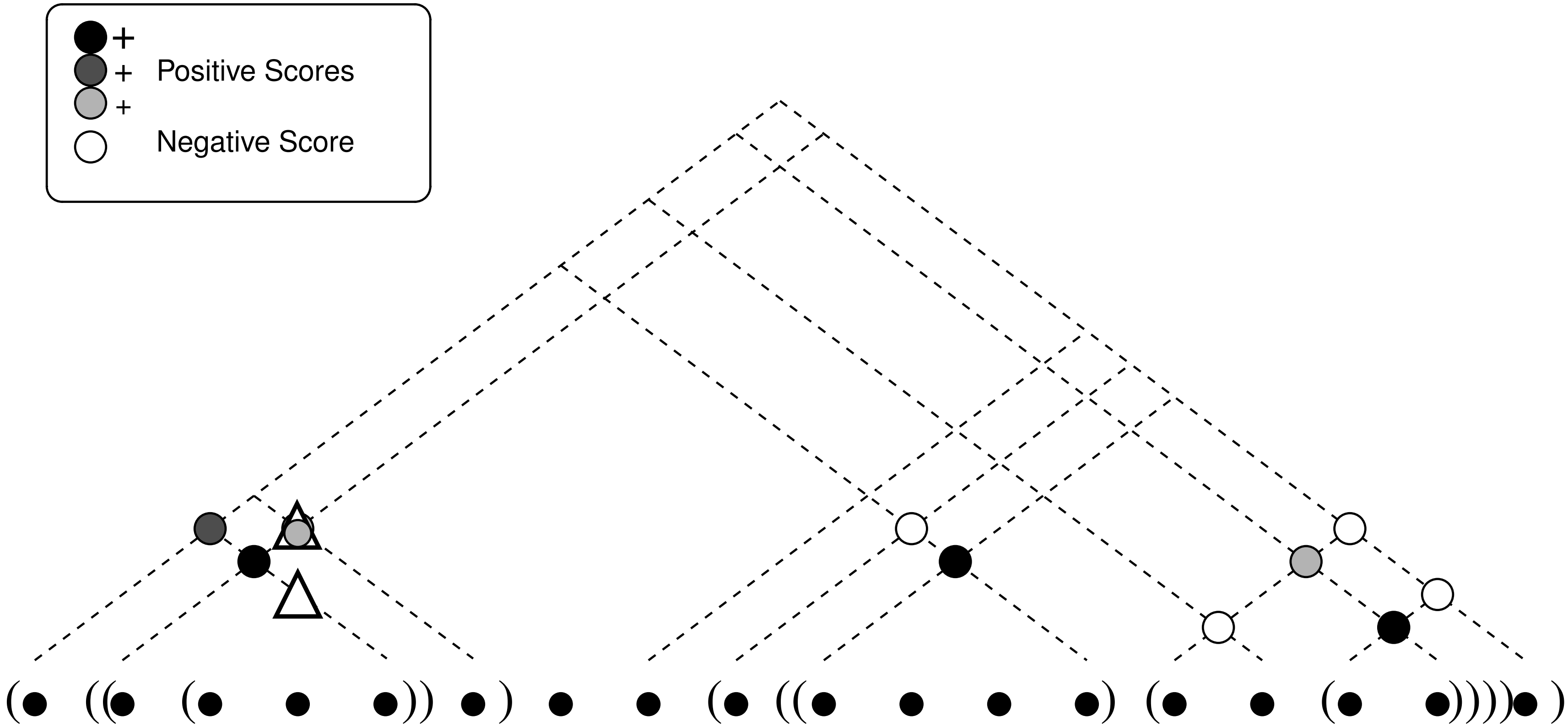
Filtering-Ranking Strategy



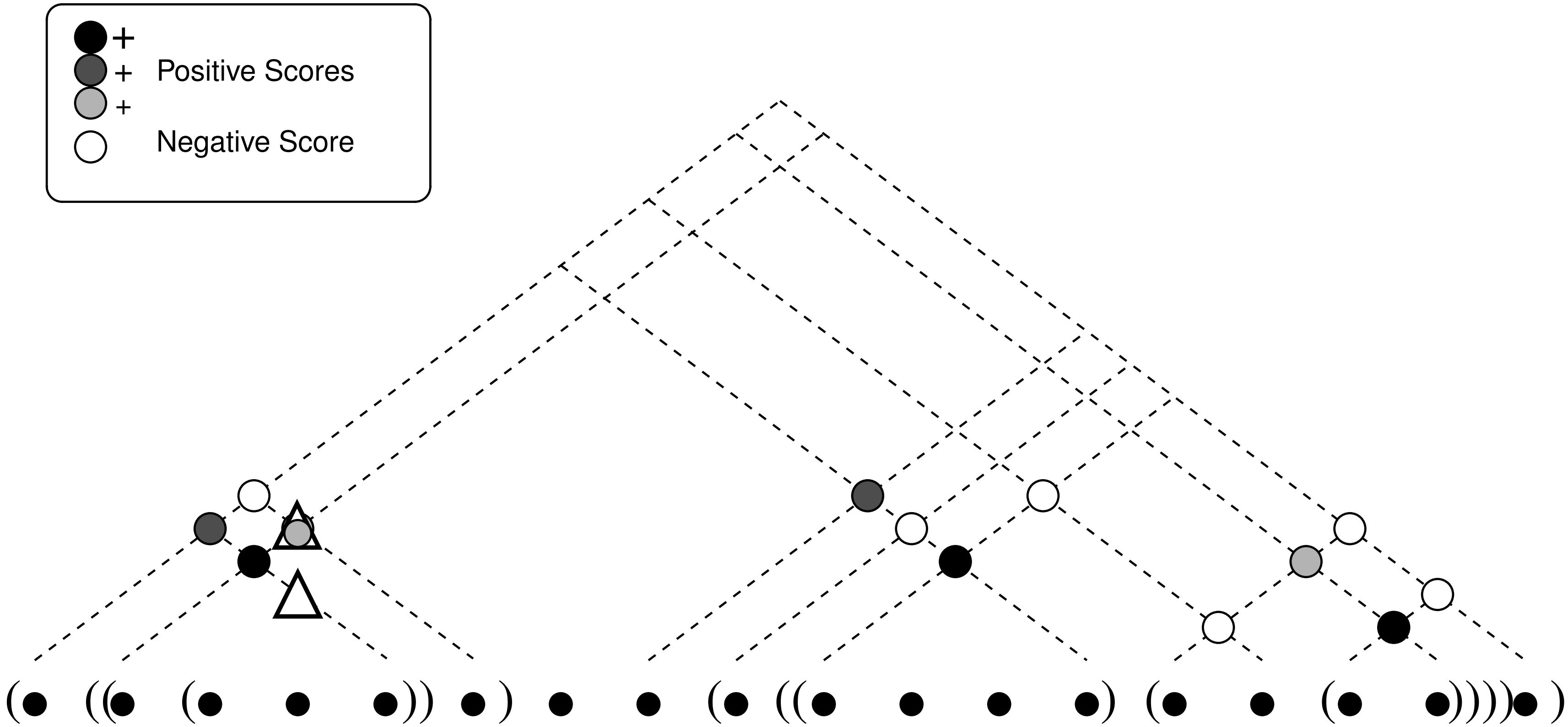
Filtering-Ranking Strategy



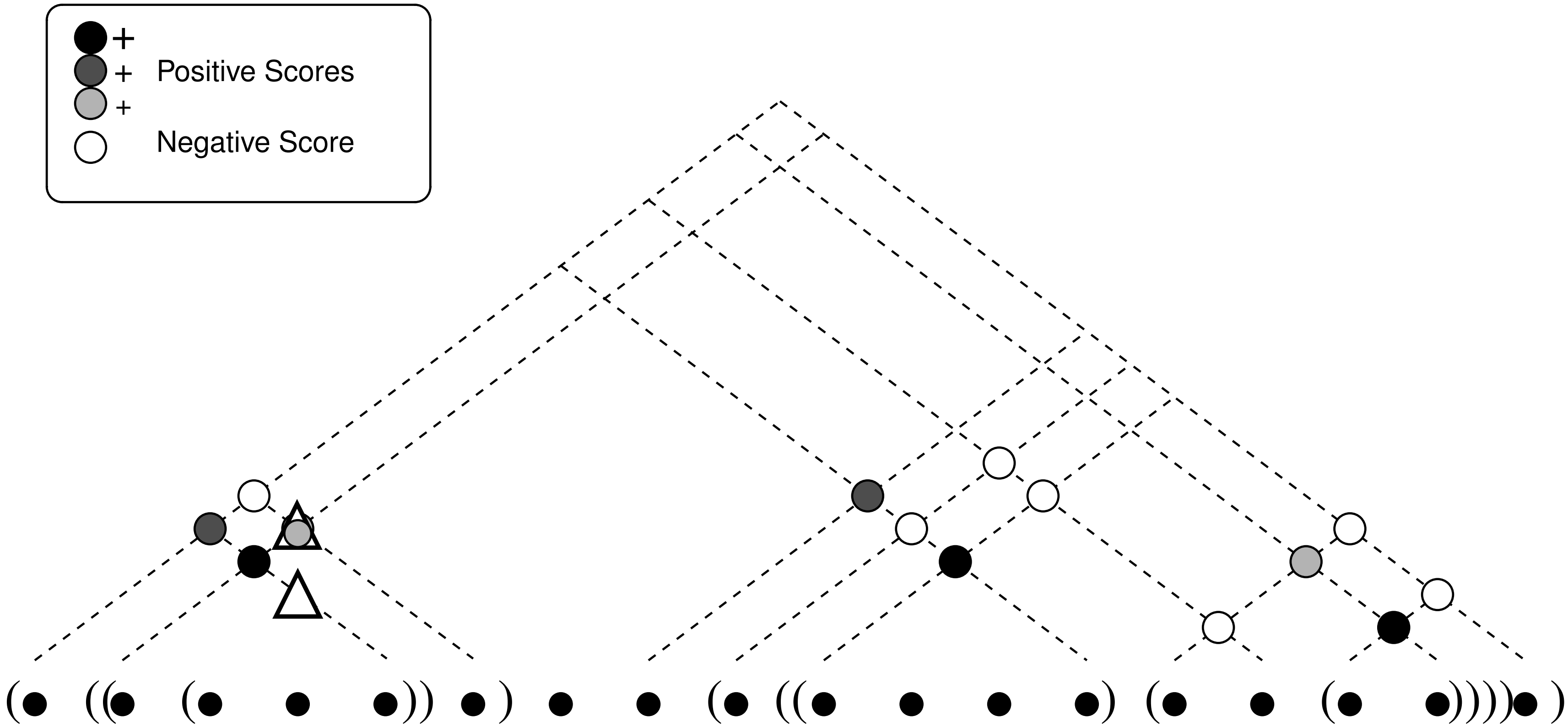
Filtering-Ranking Strategy



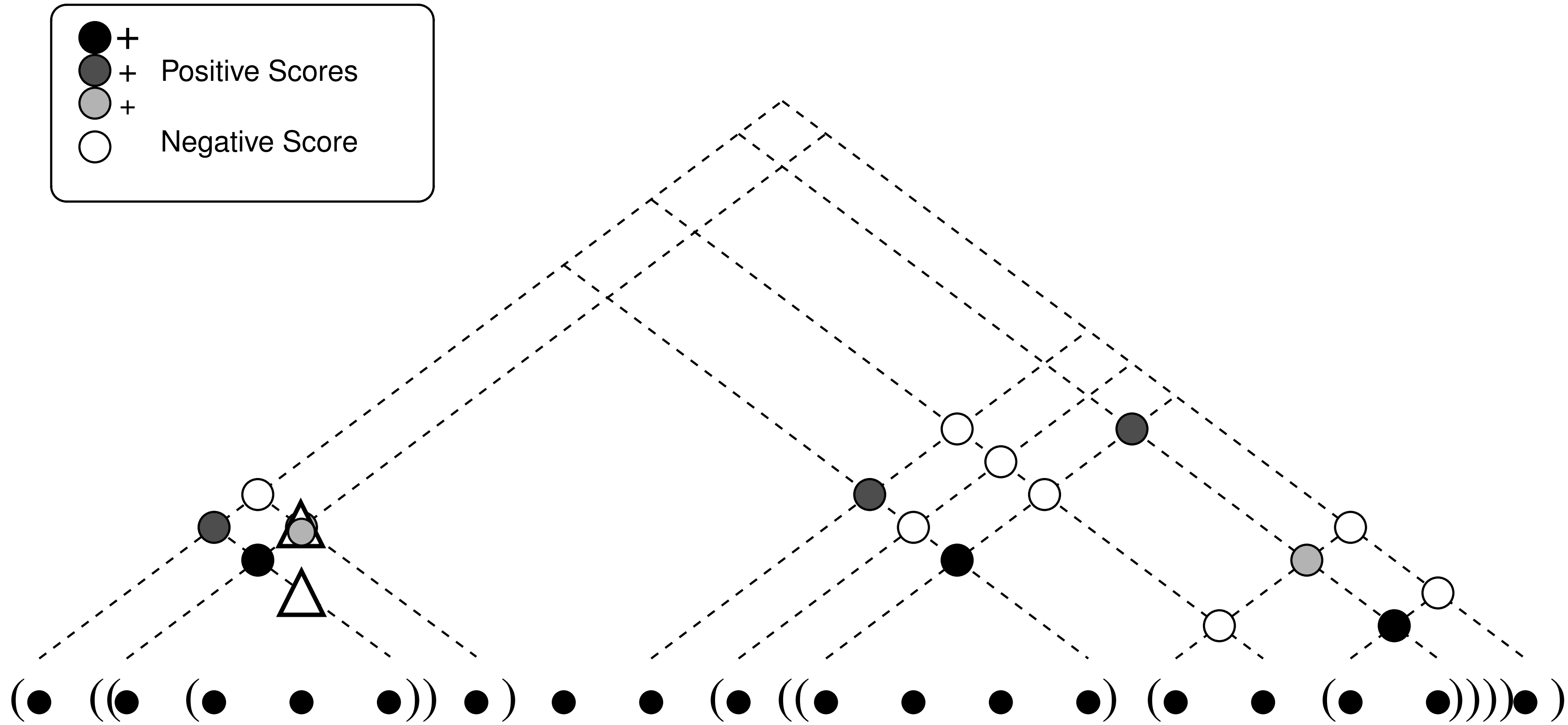
Filtering-Ranking Strategy



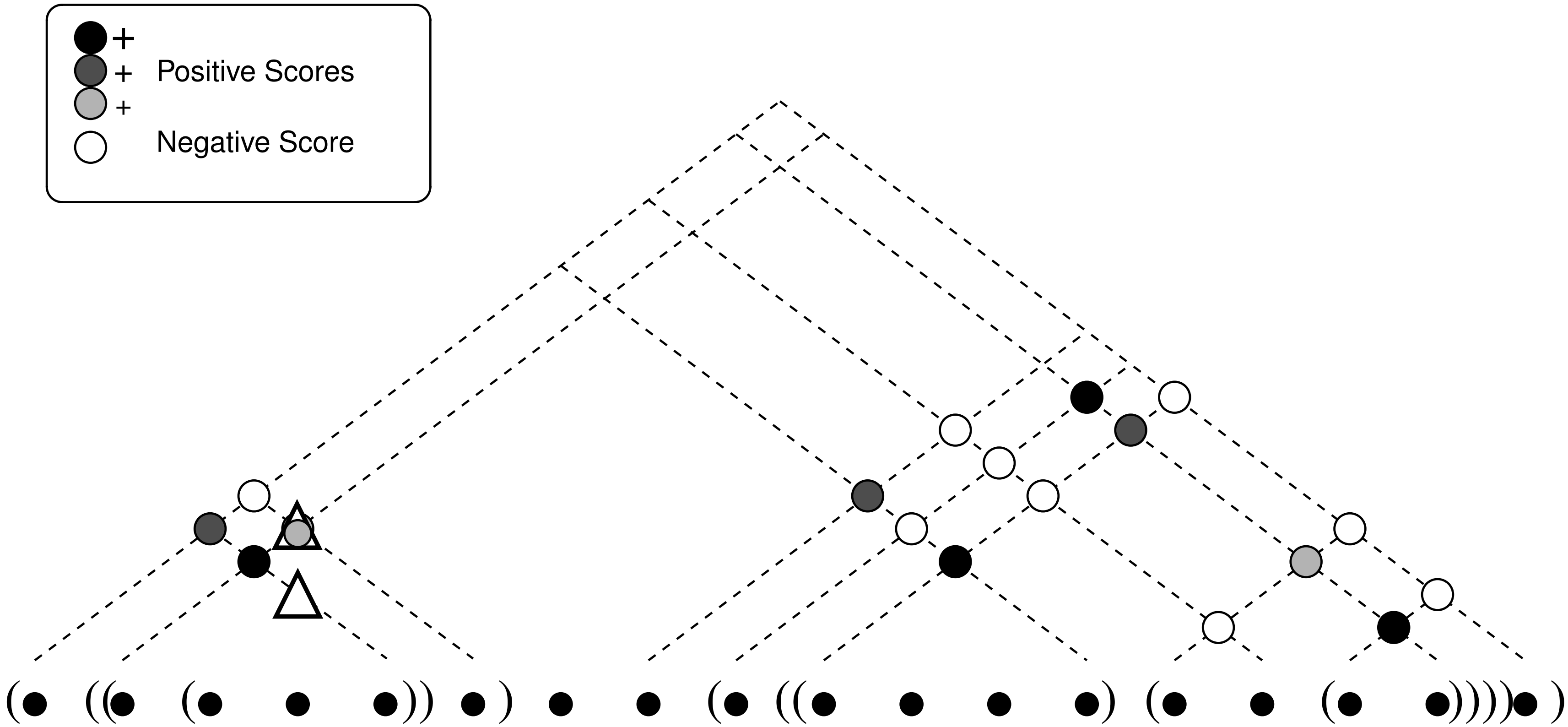
Filtering-Ranking Strategy



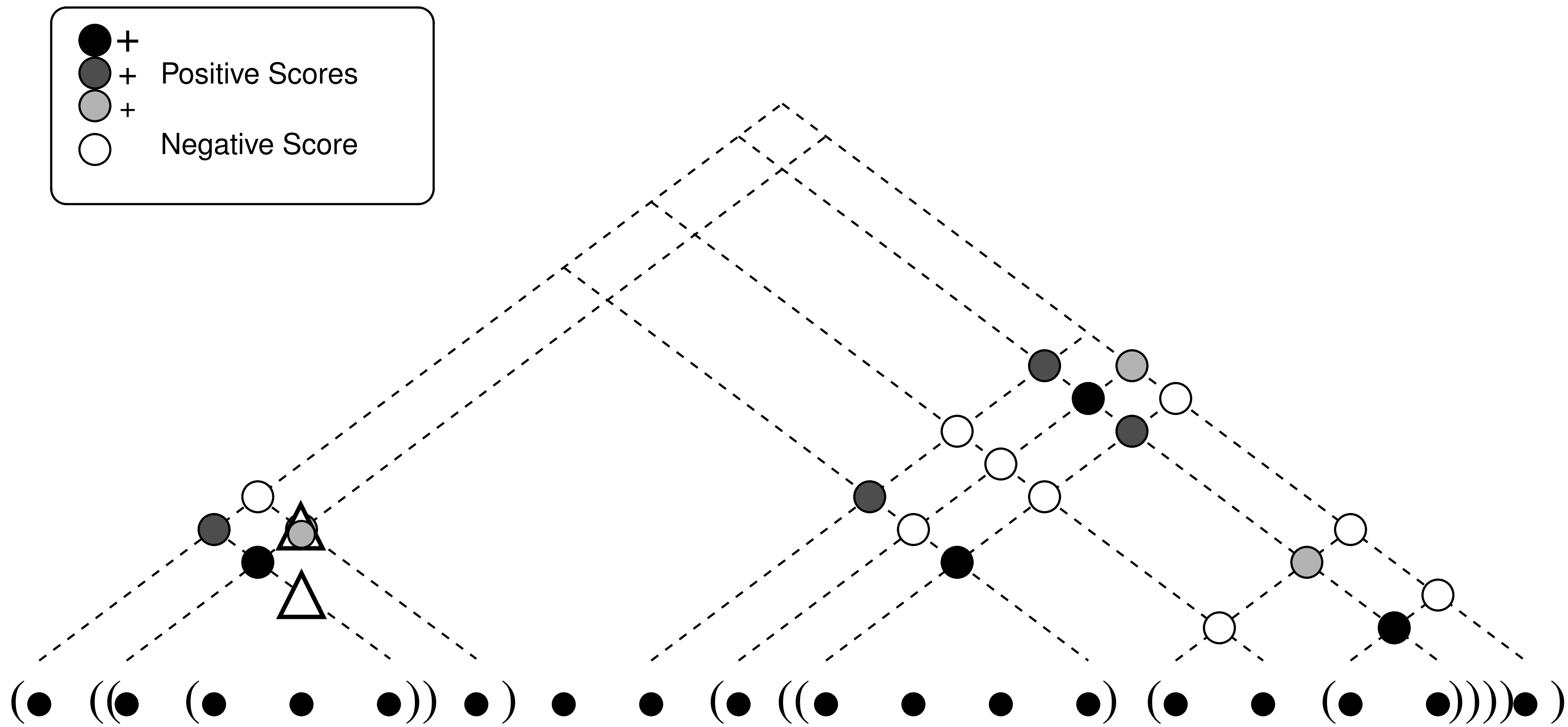
Filtering-Ranking Strategy



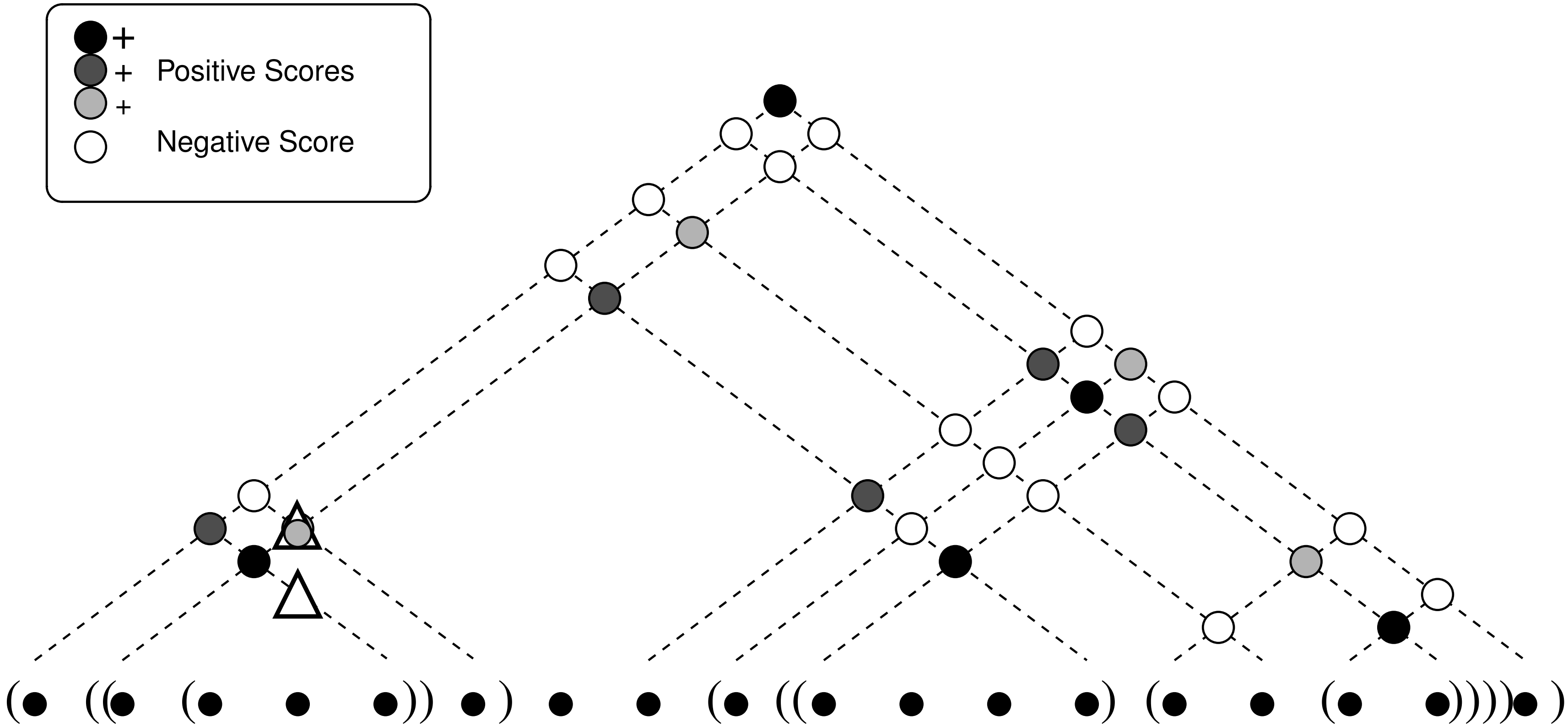
Filtering-Ranking Strategy



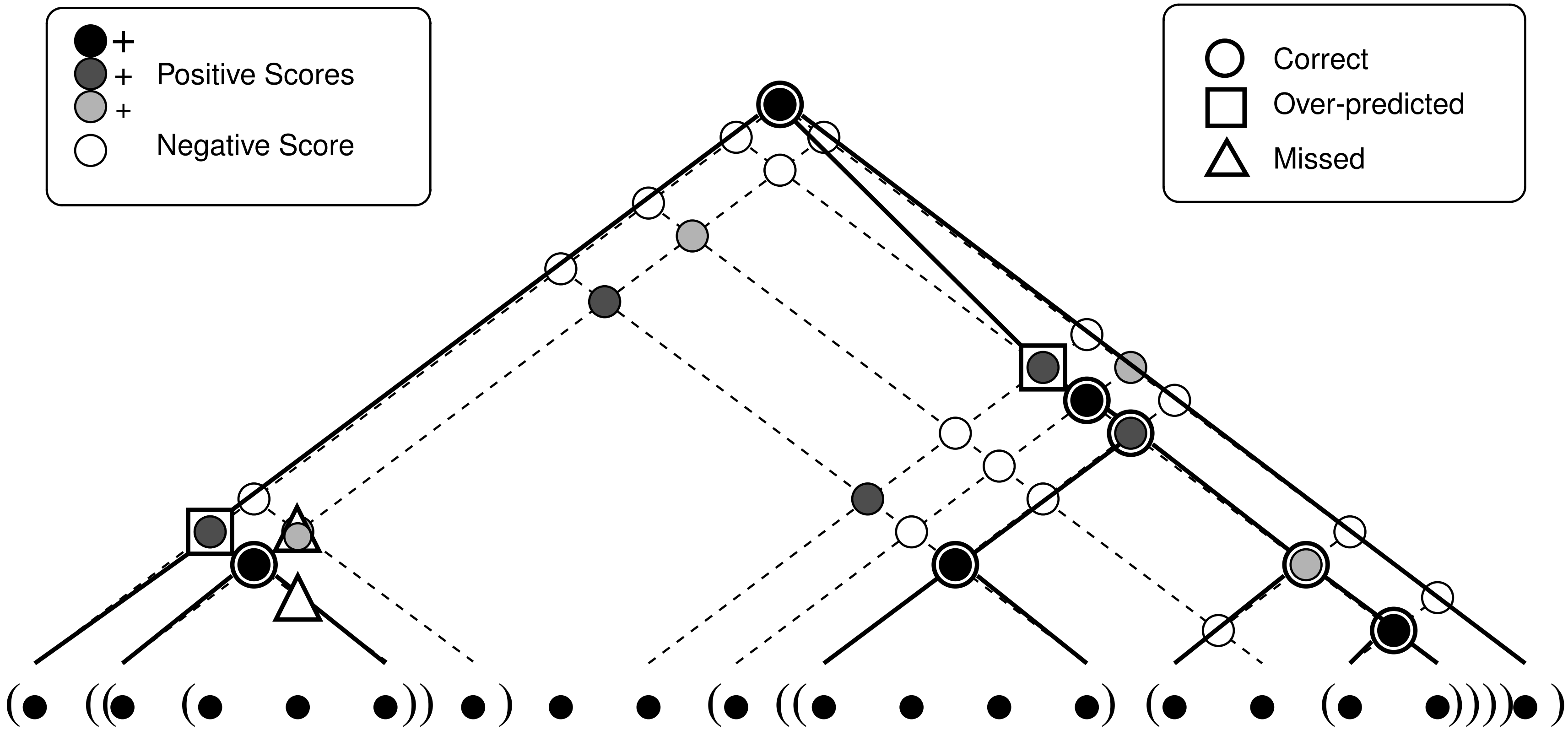
Filtering-Ranking Strategy



Filtering-Ranking Strategy



Filtering-Ranking Strategy



Learning a Filtering-Ranking Model

- Goal: Learn the functions $(\mathrm{start_w}, \mathrm{end_w}, \mathrm{score_p})$ so as to maximize the F_1 measure on the recognition of phrases

- Desired behavior:
  - ⋆ Start-End Filters:
    - ∗ Do not block any correct phrase: very high recall
    - ∗ Block phrases that produce errors at the ranking stage
    - ∗ Block much incorrect phrases as possible
  - ⋆ Ranker:
    - ∗ Separate between correct/incorrect structures
    - ∗ Forget about filtered phrases

Perceptron Learning at Global Level

- Following **[Collins 02]**, we guide learning at global level:
  - ⋆ Do not concentrate on individual errors of the learning functions
  - ⋆ Instead, concentrate on errors at sentence level, after inference

- Key points:
  - ⋆ Mistake-driven learning, a.k.a. Perceptron
  - ⋆ Functions are learned together, visiting online training sentences
  - ⋆ Errors are propagated from sentence-level, to phrase-level, to word-level

(parenthesis) Perceptron algorithm

- **[Rosenblatt, 1958]**
- Linear discriminant function, $h_{\mathbf{w}} : \mathbb{R}^n \rightarrow \mathbb{R}$, parameterized by a weight vector **w**
- Classification rule: $h_{\mathbf{w}}(\mathbf{x}) = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x}) = \hat{y}$
- On-line error-driven learning algorithm
- Additive updating rule: promotion/demotion updates

(parenthesis) Perceptron algorithm

Training set $S = \{(x_i, y_i)\}_{i=1}^{n}$ with $y_i \in \{0, 1\}$

Initialization: set parameters $\mathbf{w} = 0$
repeat for N epochs
  for $i = 1 \ldots n$
    $\hat{y} = \text{sign}(\mathbf{w} \cdot \Phi(x_i))$
    if $(y_i \neq \hat{y})$ **then** $\boxed{\mathbf{w} = \mathbf{w} + y_i \Phi(x_i)}$ **end-if**
  end-for
end-repeat
output($\mathbf{w}$)

- If data is linearly separable Perceptron converges (Novikoff): positive margin on the training set
- Extensions: dual perceptron + kernels + voted perceptron + large margin

Filtering-Ranking Perceptron

- Configuration:
  - ⋆ Feature extraction functions (given): ϕ_{w}, ϕ_{p}
  - ⋆ Weight vectors (learned): $\mathbf{w}_{\mathrm{S}}$, $\mathbf{w}_{\mathrm{E}}$, $\mathbf{w}_{\mathrm{p}}$

- Algorithm: visit online sentence-structure pairs (x, y):
  1. Infer the best phrase structure $\hat{y}$ for x
  2. Identify errors and provide feedback to weight vectors.
     We consider only errors at global level, comparing y and $\hat{y}$:
     - ⋆ Missed Phrases (those in $y \setminus \hat{y}$)
     - ⋆ Over-predicted Phrases (those in $\hat{y} \setminus y$)

FR-Perceptron: Feedback on Missed phrases

If a phrase $(s, e)_k$ is missed, do promotion updates:

- if word s is not positive start for k:
$$\mathbf{w}_{\mathrm{S}} = \mathbf{w}_{\mathrm{S}} + \phi_{\mathrm{w}}(x, s, k)$$
- if word e is not positive end for k:
$$\mathbf{w}_{\mathrm{E}} = \mathbf{w}_{\mathrm{E}} + \phi_{\mathrm{w}}(x, e, k)$$
- if $(s, e)_k$ passes the filter (s/e are positive start/end for k):
$$\mathbf{w}_{\mathrm{p}} = \mathbf{w}_{\mathrm{p}} + \phi_{\mathrm{p}}(x, y, (s, e)_k)$$

FR-Perceptron: Feedback on Over-Predicted phrases

If a phrase $(s, e)_k$ is over-predicted, do demotion updates:

- Give feedback to the ranker:

$$\mathbf{w}_{\mathrm{p}} = \mathbf{w}_{\mathrm{p}} - \phi_{\mathrm{p}}(x, y, (s, e)_k)$$

- If word s is not a correct start for k:

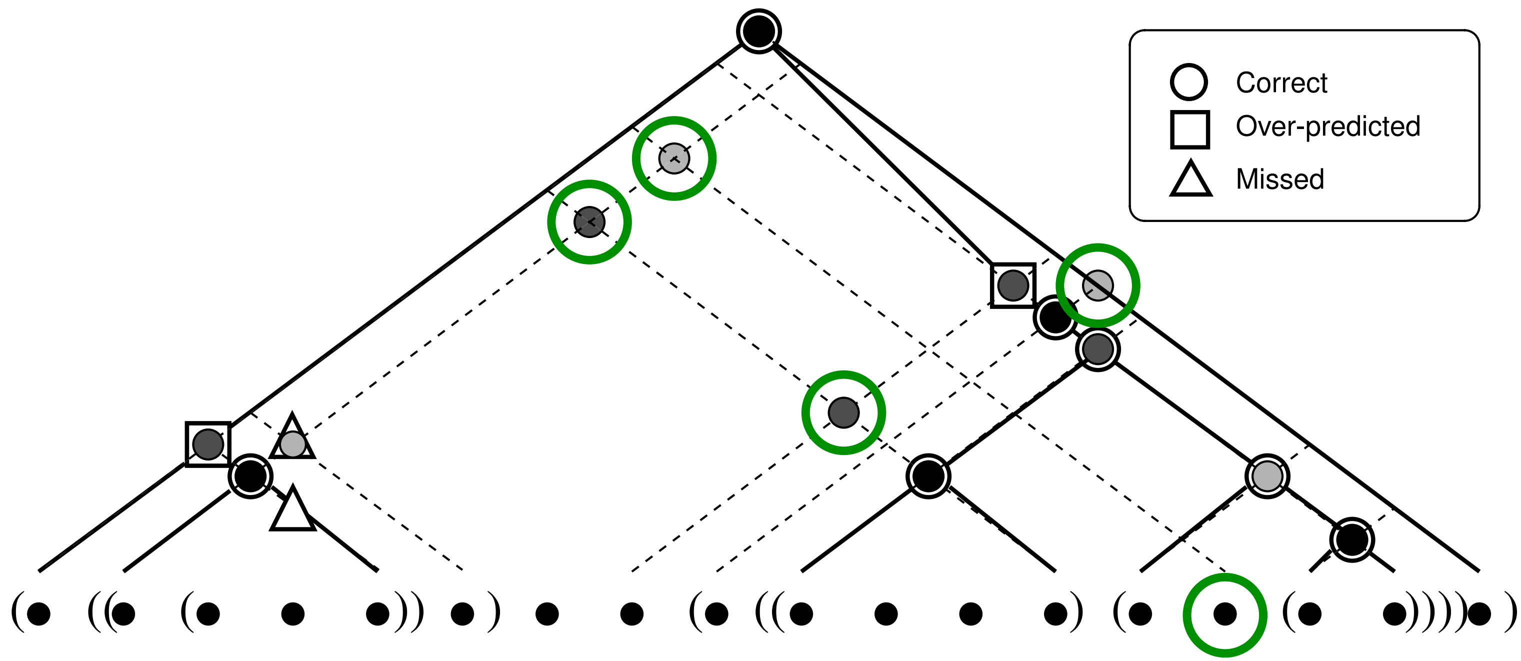
$$\mathbf{w}_{\mathrm{S}} = \mathbf{w}_{\mathrm{S}} - \phi_{\mathrm{w}}(x, s, k)$$

- If word e is not a correct end for k:

$$\mathbf{w}_{\mathrm{E}} = \mathbf{w}_{\mathrm{E}} - \phi_{\mathrm{w}}(x, e, k)$$

Learning Feedback: Example

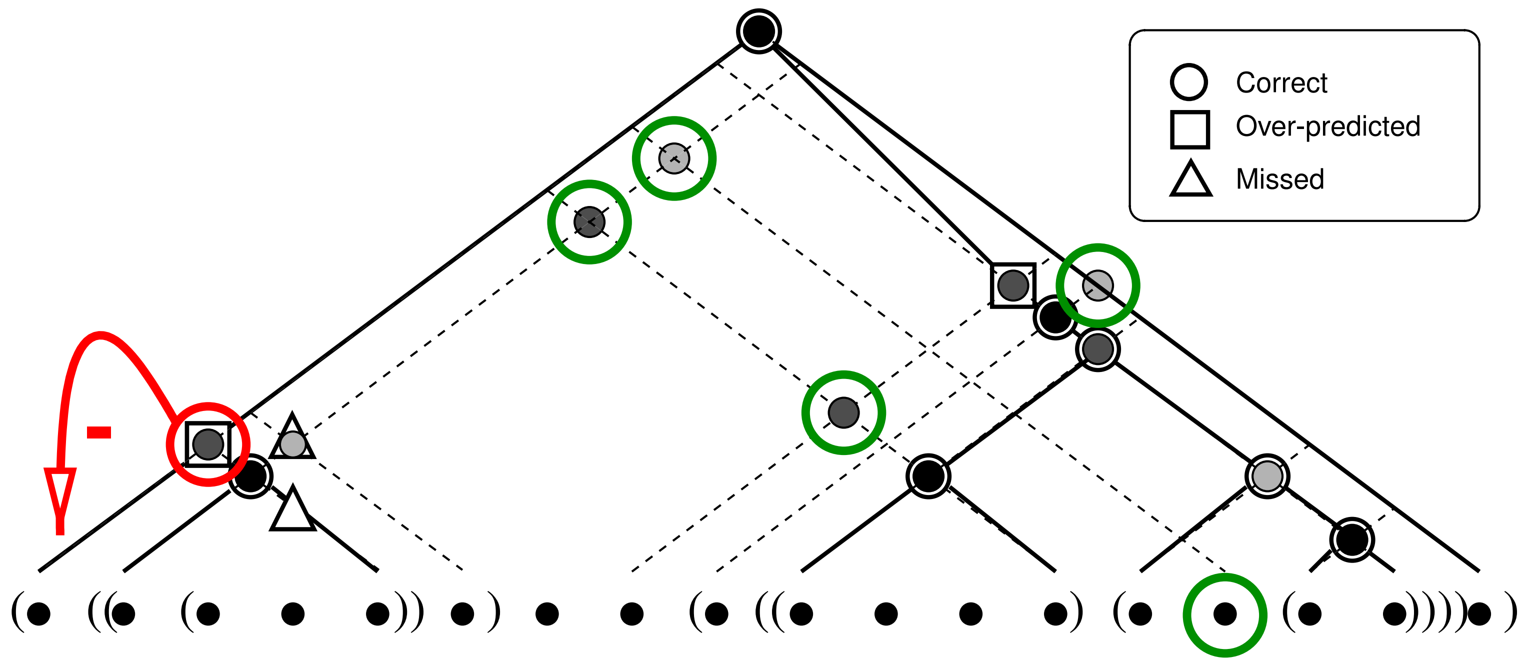
- Local predictions are **corrected** wrt. the global solution



- Local predictions that do not hurt globally are **not penalized**

Learning Feedback: Example

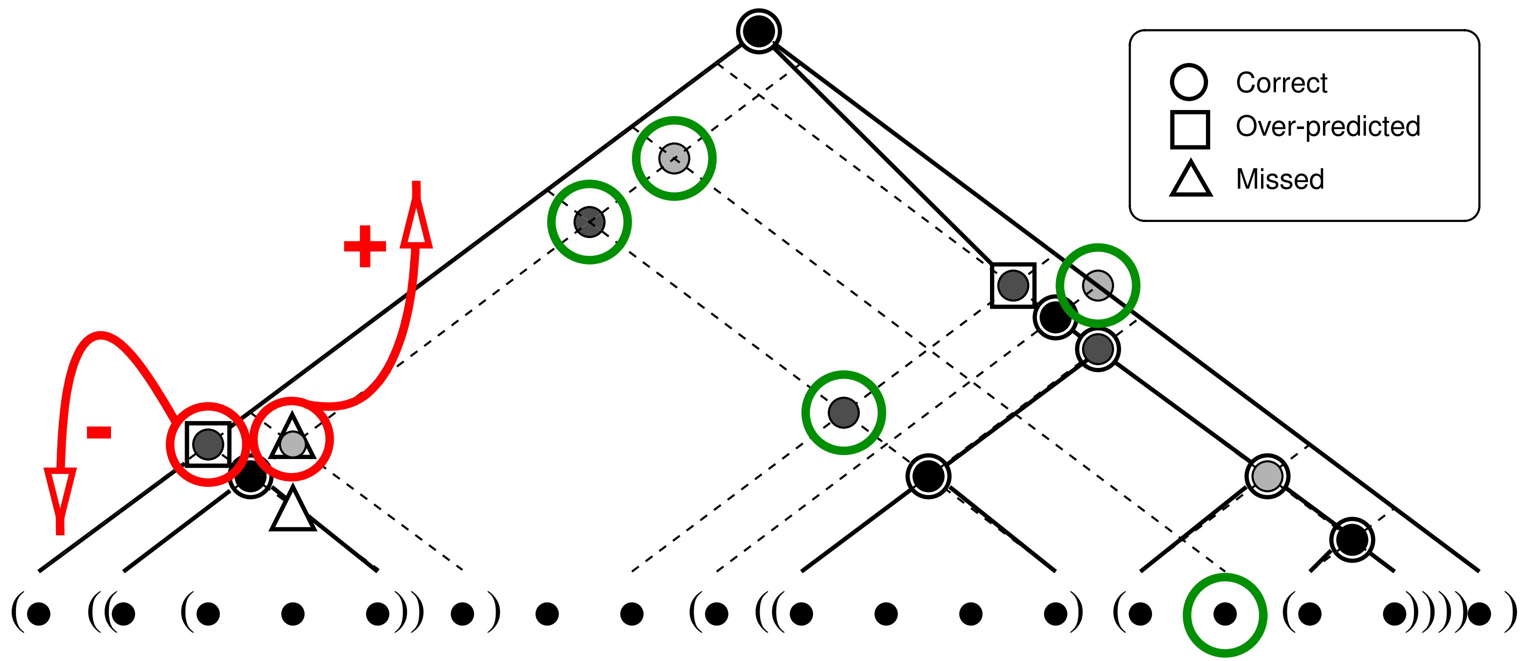
- Local predictions are **corrected** wrt. the global solution



- Local predictions that do not hurt globally are **not penalized**

Learning Feedback: Example

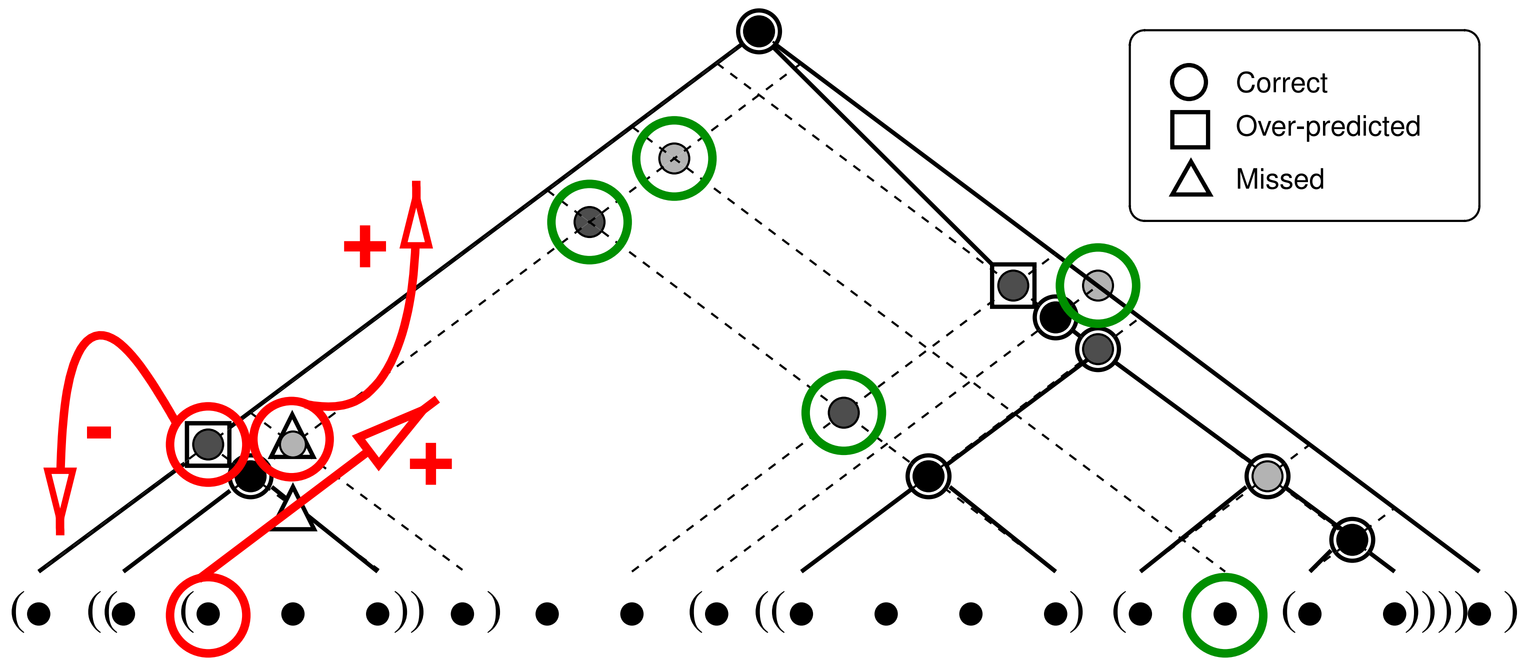
- Local predictions are **corrected** wrt. the global solution



- Local predictions that do not hurt globally are **not penalized**

Learning Feedback: Example

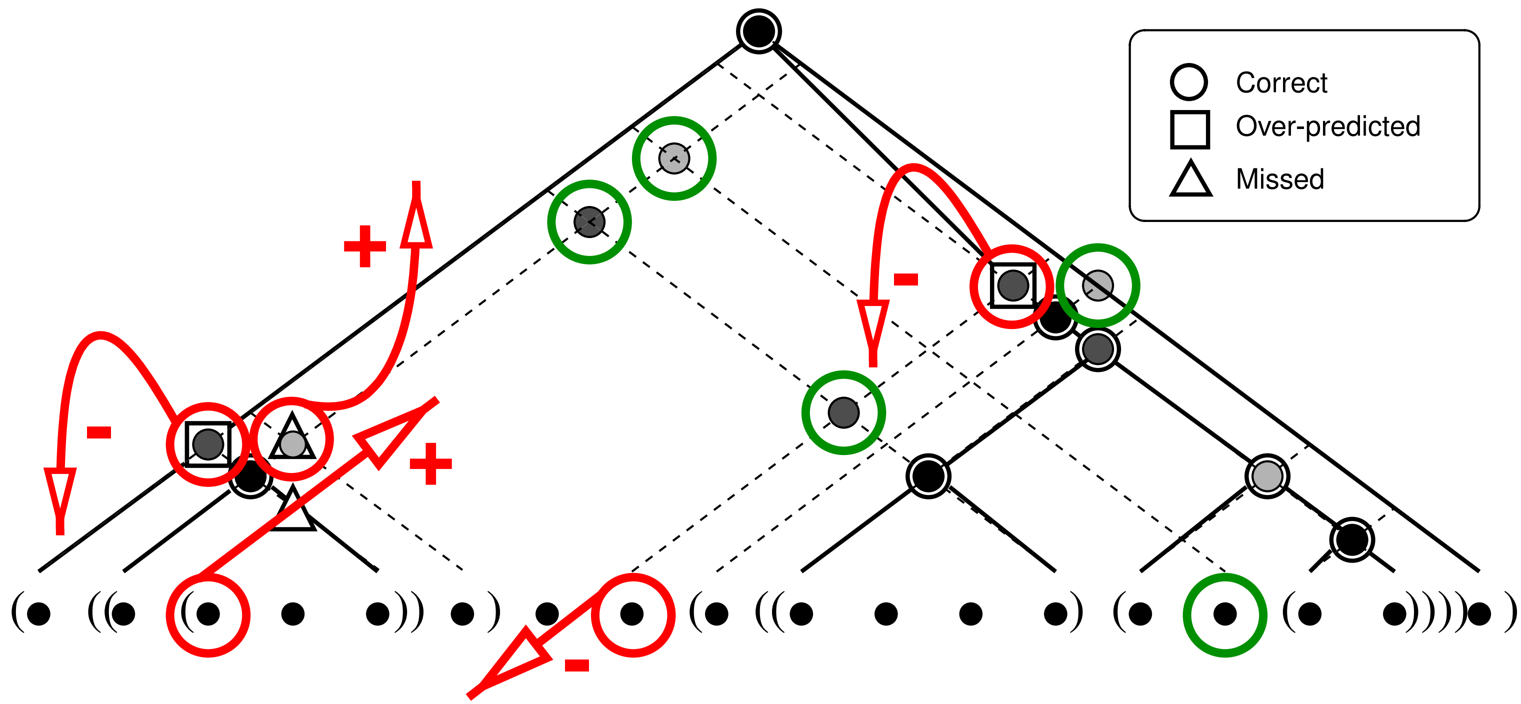
- Local predictions are **corrected** wrt. the global solution



- Local predictions that do not hurt globally are **not penalized**

Learning Feedback: Example

- Local predictions are **corrected** wrt. the global solution



- Local predictions that do not hurt globally are **not penalized**

Empirical validation of FR-Perceptron

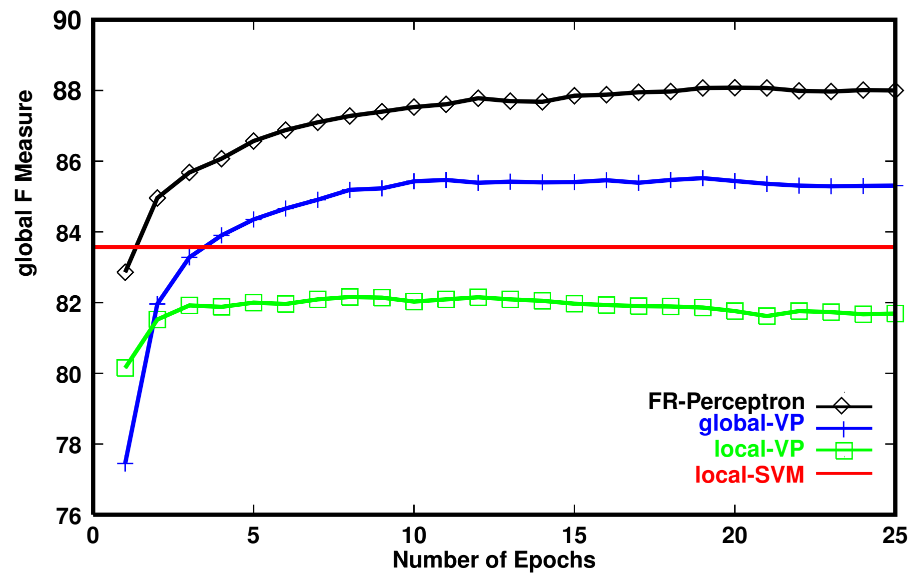
- We perform a number of experiments to validate the behavior of FR-Perceptron
- Problem: Clause Identification, following CoNLL-2001 Shared Task:
  - ⋆ One type of phrases: clauses
  - ⋆ Hierarchical Structure
  - ⋆ Training: $\sim$ 9,000 sentences, $\sim$ 25,000 clauses
  - ⋆ Test: $\sim$ 1,700 sentences, $\sim$ 4,900 clauses

Empirical validation of FR-Perceptron

We compare four training strategies for the Filtering-Ranking model:

	type	**w's trained**	**R on F**	**penalty wrt.**
local-VP	VP	separatedly	no	binary sign
local-SVM	SVM	separatedly	no	binary sign
global-VP	VP	together	yes	binary sign
FR-Perceptron	VP	together	yes	$\arg\max$

Empirical validation of FR-Perceptron
Overall Results



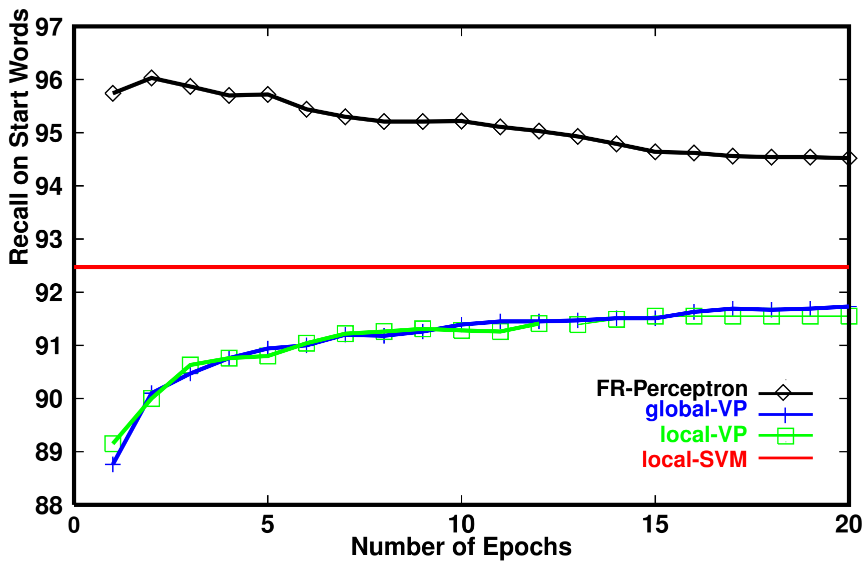
- Global training strategies perform better than local strategies
- Feedback after inference trains more effectively the recognizer

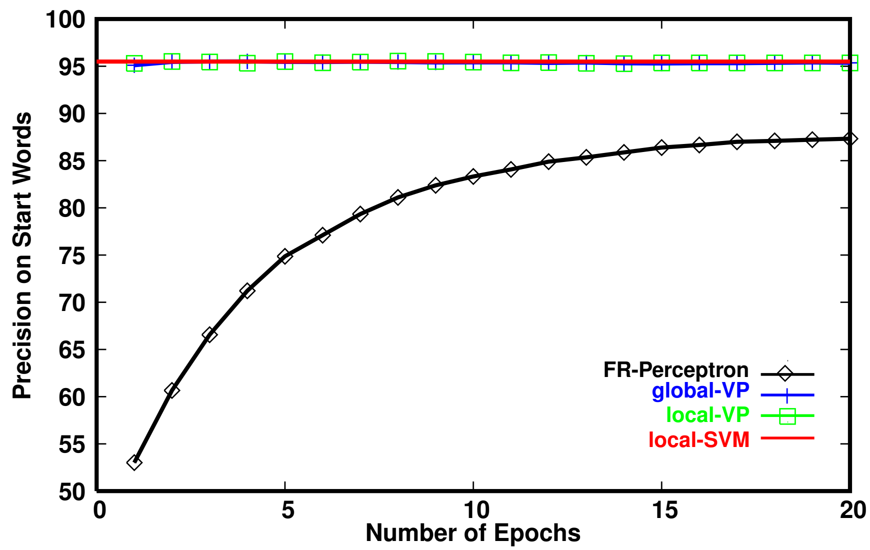
Empirical validation of FR-Perceptron Behavior of the Start-End Filter

We look at the performance of Start-End functions:

- Precision/Recall of Start-End
- How much the phrase space is reduced?
- What is the maximum achievable F_1 after the Filter?

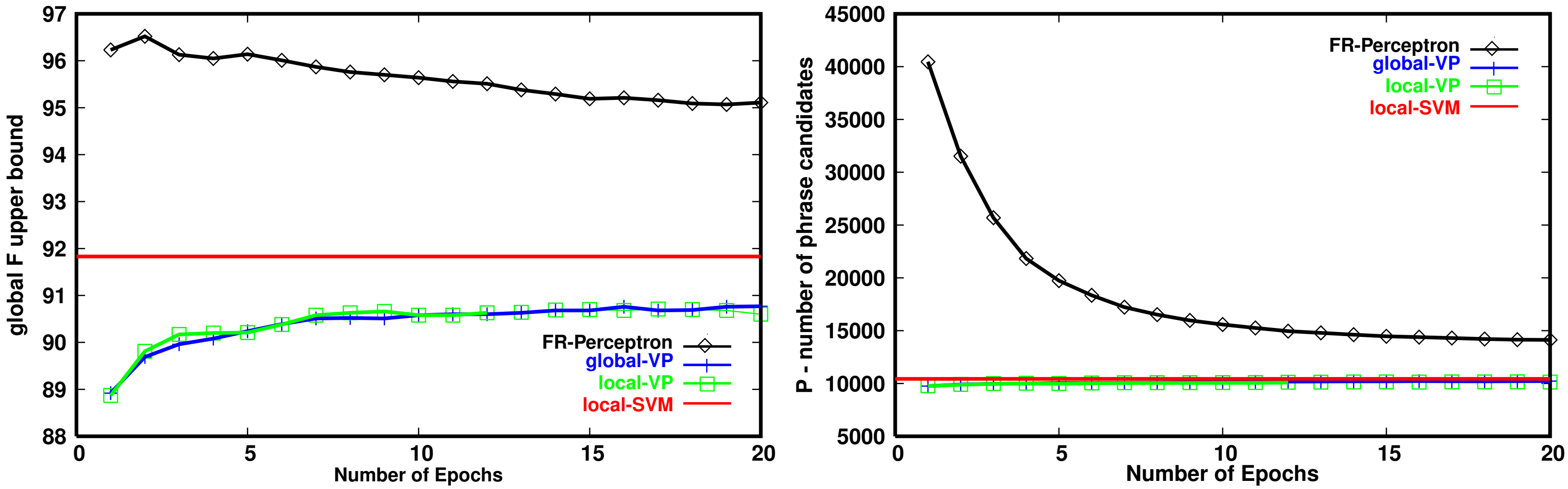
Empirical validation of FR-Perceptron Recall/Precision on Start words





- FR-Perceptron favors recall, others favor precision
- On End words, the same behavior is observed

Experiments on Clause Identification Upper Bound F_1/Explored Phrases

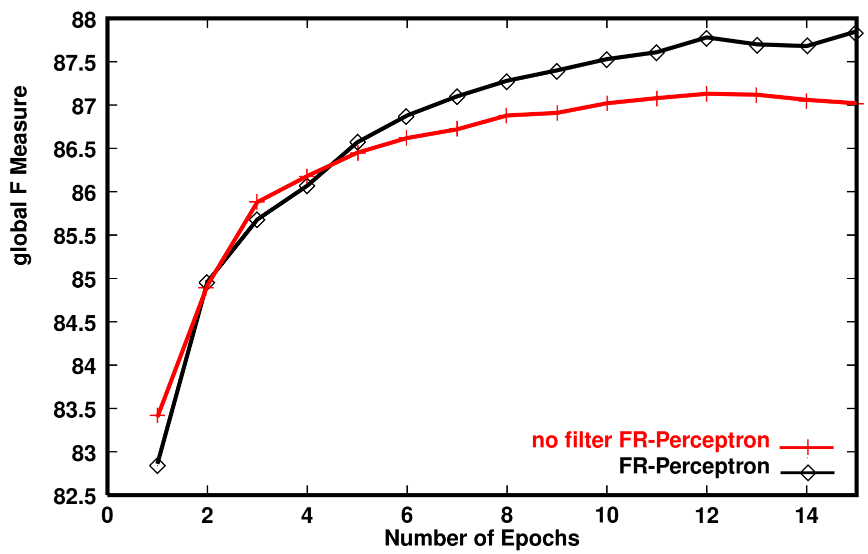


- FR-Perceptron maintains a high upper-bound F_1 for the ranking layer (left), and reduces the space of explored phrases (right)

- Other methods are not sensitive to F-R interactions

Empirical validation of FR-Perceptron Does the Filter help in performance?

- We train the architecture without the Filter (UB-$F_1 = 100\%$):



- Filtering favors not only efficiency, but also global accuracy

Course Overview

- Introduction and Motivation
  - ⋆ NLP tasks and Learning & NLP tools in the Web
- Machine Learning for supervised classification
  - ⋆ Machine learning algorithms & Applications
- **Statistical Learning for Structured NLP**
  - ⋆ Generative models
  - ⋆ The Learning and Inference paradigm
  - ⋆ Re-ranking candidate solutions
  - ⋆ Structure learning with global linear models
- A case study: Semantic Role Labeling

(Reranking) A reminder of the learning setting

- $\mathcal{X}$ is a set of possible inputs
- $\mathcal{Y}$ is a set of possible outputs
- A training set $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where each $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$
- We assume that S is generated i.i.d. from an unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$
- Goal is to learn a hypothesis function $F : \mathcal{X} \rightarrow \mathcal{Y}$, that minimizes error on the entire distribution $\mathcal{D}$
- E.g., each x_i is a sentence, each y_i is a gold-standard parse

Reranking: Setting

- Three components:
  - $\star$ $\textsc{Gen}$ is a function from a string to a set of candidates
  - $\star$ Φ maps a candidate to a feature vector
  - $\star$ score$(\Phi, x, \hat{y})$ a function that scores the appropriateness of candidate solution $\hat{y}$ for input example x.

- F is of the form: $F(x) = \arg\max_{\hat{y} \in GEN(x)} \mathrm{score}(\Phi, x, \hat{y})$

- For linear classifiers: $F(x) = \arg\max_{\hat{y} \in GEN(x)} \Phi(x, \hat{y}) \cdot \mathbf{w}$, where $\mathbf{w}$ is a parameter vector.

  I.e., **Choose the highest scoring tree as the most plausible structure**

Reranking: Setting

- In reranking the function $\textsc{Gen}$ gives a set of explicit $\hat{y}$ candidates for each example x, e.g., the list of n-best parses for a sentence x produced by a statistical parser.

- Goal of learning:
  - ⋆ Given $\{(x_i, y_i)\}_{i=1}^{n}$, $\textsc{Gen}$, Φ
  - ⋆ How to set **w**?
    to make $\Phi(x, y) \cdot \mathbf{w} \geq \Phi(x, \hat{y}) \cdot \mathbf{w}$, for all candidate solutions $\hat{y}$ produced by $\textsc{Gen}$

- Reranking is an instance of metalearning

Reranking: training algorithms

- Several algorithms exists for the ranking problem, as variants of the standard learning algorithms: perceptron, boosting, SVMs, etc.
- Instead of correct classification, the learning constraint imposed by ranking (and to be consideded in the objective function) is:

$$\boxed{\forall \hat{y} \in GEN(x),\ \hat{y} \neq y : \quad \text{score}(\Phi, x, y) > \text{score}(\Phi, x, \hat{y})}$$

Reranking: perceptron learning

[Collins and Duffy, 2002]

$S = \{(x_i, y_i)\}_{i=1}^{n}$; assume $GEN(x_i) = \{y_{i1}, \ldots, y_{in_i}\}$

Initialization: set parameters $\mathbf{w} = 0$

repeat for N epochs

for $i = 1 \ldots n$

$j = \arg\max_{j=1 \ldots n_i} \Phi(x_i, y_{ij}) \cdot \mathbf{w}$

if $(y_i \neq y_{ij})$ **then** $\mathbf{w} = \mathbf{w} + \Phi(x_i, y) - \Phi(x_i, y_{ij})$

end-for

end-repeat

output: $\mathbf{w}$

- A simple extension to dual perceptron exists: kernels and voted perceptron can be used

Reranking: Max-margin learning

[Bartlett et al., 2004]

- An iterative algorithm for training a ranking function (based on EG, Exponentiated Gradient optimization)
- Learning bias is to maximize the margin of the training set (SVM-like)

Reranking: Max-margin learning

[Bartlett et al., 2004]

- An iterative algorithm for training a ranking function (based on EG, Exponentiated Gradient optimization)
- Learning bias is to maximize the margin of the training set (SVM-like)
- Specific loss functions can be set
- The algorithm converges to the exact solution
- See slides from Michael Collins' presentation at CoNLL-2006 (PDF file)

Reranking

A couple of technical and practical questions

- How to generate the training set for learning the ranker?
- What if the gold standard y is not among the candidates $\hat{y}$?

Reranking: applications

- Parse reranking
  [Johnson et al, 1999; Collins, 2000; Shen, Sarkar and Joshi, 2003; Riezler et al., 2004; Charniak and Johnson, 2005; Collins and Koo, 2005]
- Reranking for Machine Translation
  [Shen, Sarkar and Och, 2004; Shen and Joshi, 2005]
- Semantic Role Labeling **[Haghighi et al., 2005]**

Reranking: pros & cons

Pros

- Rich complex features can be designed on the complete structure. This may facilitate capturing long-distance dependencies among substructures
- Simple (and efficient) learning algorithms exist

Cons

- Dependence on the base linguistic processor implementing GEN. High recall must be ensured with a few candidates. The theoretical upper bounds on task accuracy can be substantially lowered
- The two-step procedure can make the system less efficient

Course Overview

- Introduction and Motivation
  - ⋆ NLP tasks and Learning & NLP tools in the Web
- Machine Learning for supervised classification
  - ⋆ Machine learning algorithms & Applications
- **Statistical Learning for Structured NLP**
  - ⋆ Generative models
  - ⋆ The Learning and Inference paradigm
  - ⋆ Re-ranking candidate solutions
  - ⋆ Structure learning with global linear models
- A case study: Semantic Role Labeling

Course Overview

- ...
- **Statistical Learning for Structured NLP**
  - ⋆ Generative models
  - ⋆ The Learning and Inference paradigm
  - ⋆ Re-ranking candidate solutions
  - ⋆ Structure learning with global linear models
    - ∗ Perceptron Global Learning
    - ∗ Max-margin learning algorithms
    - ∗ Conditional Random Fields
- ...

Structure learning with global linear models

- Same setting as before but considering all possible candidates
- In this case the size of $GEN(x)$ is exponential w.r.t. the length of x and cannot be treated explicitely
- Dynamic programming is used to perform inference (i.e., the calculation of $\arg\max$) —sometimes inference is only approximate. $\Phi(x, \hat{y})$ and $L(x, y, \hat{y})$ have to decompose consistently with the inference algorithm. Local versions of feature-codification and loss functions are considered $\phi(x, r)$ and $l(x, y, r)$.
- Learning and inference are usually coupled by using on-line learning algorithms

Structure learning with global linear models

- Several models/training-algorithms exist:
  - ⋆ Perceptron global learning for a sequential Markov model **[Collins 2002]**
  - ⋆ **Max-Margin Markov Networks** **[Taskar et al., 2003]** and an EG-based training algorithm **[Bartlett, et al., 2004]**
  - ⋆ **Hidden Markov Support Vector Machines** **[Altun et al., 2003]** and a training algorithm **[Tsochantaridis et al., 2004]** implemented and publicly available at the $SVMstruct$ suite.
  - ⋆ Probabilistic approaches: **Conditional Random Fields** **[Lafferty, et al., 2001; Sha and Pereira, 2002]**

Course Overview

- ...
- **Statistical Learning for Structured NLP**
  - ⋆ Generative models
  - ⋆ The Learning and Inference paradigm
  - ⋆ Re-ranking candidate solutions
  - ⋆ Structure learning with global linear models
    - ∗ Perceptron Global Learning
    - ∗ Max-margin learning algorithms
    - ∗ Conditional Random Fields
- ...

Learning a global ranker: Perceptron training

[Collins, 2002]

- Presents a Perceptron learning algorithm for discriminatively training a Markov-based tagger
- Global ranking approach: all candidate solutions are considered
- Instantiation on sequential labeling problems using Viterbi-like inference
- Improved results, on POS tagging and NP chunking, compared to a MaxEnt tagger
- Best paper award at EMNLP-2002

(recall) Reranking: perceptron learning

[Collins and Duffy, 2002]

$S = \{(x_i, y_i)\}_{i=1}^{n}$; assume $GEN(x_i) = \{y_{i1}, \ldots, y_{in_i}\}$

Initialization: set parameters $\mathbf{w} = 0$
repeat for N epochs
  for $i = 1 \ldots n$
    $j = \arg\max_{j=1 \ldots n_i} \Phi(x_i, y_{ij}) \cdot \mathbf{w}$
    if $(y_i \neq y_{ij})$ **then** $\mathbf{w} = \mathbf{w} + \Phi(x_i, y) - \Phi(x_i, y_{ij})$
  end-for
end-repeat
output: $\mathbf{w}$

- A simple extension to dual perceptron exists: kernels and voted perceptron can be used

Learning a global ranker: Perceptron training

[Collins, 2002]

Initialization: set parameters $\mathbf{w} = 0$

repeat for N epochs

  for $i = 1 \ldots n$

    $\hat{y}_i = \arg\max_{y \in GEN(x_i)} \Phi(x_i, y) \cdot \mathbf{w}$

    if $(y_i \neq \hat{y}_i)$ **then** $\mathbf{w} = \mathbf{w} + \Phi(x_i, y) - \Phi(x_i, \hat{y}_i)$

  end-for

end-repeat

output: $\mathbf{w}$

- The general algorithm is the same, but now GEN generates all possible solutions
- $|\text{GEN}(x)|$ is exponential in the length of x
- The implementation cannot list all $y \in GEN(x_i)$ explicitely

Decomposition: Local/Global

[Collins, 2002]

- Restricted to sequential labeling problems:
  - ⋆ $x = x_1, x_2, \ldots, x_n = x_{[1:n]} = \mathbf{x}$
  - ⋆ $y = t_1, t_2, \ldots, t_n = t_{[1:n]} = \mathbf{t}$
- "$\arg\max$" inference is performed using Viterbi search
- Each example $(\mathbf{x}, \mathbf{t})$ is decomposed into a finite set of parts $R(\mathbf{x}, \mathbf{t}) \subseteq \mathcal{R}$.
- In a trigram based tagger (linear number of parts): $R(\mathbf{x}, \mathbf{t}) = \bigcup_{i=1}^{n} r_i = \bigcup_{i=1}^{n} \langle t_i, t_{i-1}, t_{i-2}, \mathbf{x}, i \rangle$

Decomposition: Feature Vectors

[Collins, 2002]

- Local feature-vector representation:
  $\phi(r_i) = \phi(\langle t_i, t_{i-1}, t_{i-2}, \mathbf{x}, i\rangle) = (\phi_1(r_i), \phi_2(r_i), \ldots, \phi_d(r_i))$
  with local feature functions $\phi_j(r_i)$, one for each dimension

Decomposition: Feature Vectors

[Collins, 2002]

- Local feature-vector representation:
  $\phi(r_i) = \phi(\langle t_i, t_{i-1}, t_{i-2}, \mathbf{x}, i\rangle) = (\phi_1(r_i), \phi_2(r_i), \ldots, \phi_d(r_i))$
  with local feature functions $\phi_j(r_i)$, one for each dimension
- The global feature-vector representation is the sum of local representations: $\boxed{\Phi(x_{[1:n]}, t_{[1:n]}) = \sum_{i=1}^{n} \phi(r_i)}$

Decomposition: Feature Vectors

[Collins, 2002]

- Local feature-vector representation:
  $\phi(r_i) = \phi(\langle t_i, t_{i-1}, t_{i-2}, \mathbf{x}, i \rangle) = (\phi_1(r_i), \phi_2(r_i), \ldots, \phi_d(r_i))$
  with local feature functions $\phi_j(r_i)$, one for each dimension
- The global feature-vector representation is the sum of local representations: $\boxed{\Phi(x_{[1:n]}, t_{[1:n]}) = \sum_{i=1}^{n} \phi(r_i)}$
- If local features are indicator functions, then the global features will be "counts", e.g:

$$\phi_{1000}(r_i) = \begin{cases} 1 & \text{if current word } x_i \text{ is "the" and } t_i = \text{DT} \\ 0 & \text{otherwise} \end{cases}$$

Then, $\Phi_{1000}(x_{[1:n]}, t_{[1:n]}) = \sum_{i=1}^{n} \phi_{1000}(r_i)$ is the number of times "the" is seen tagged as DT in the example $(x_{[1:n]}, t_{[1:n]})$

Learning a global ranker: objective function

[Collins, 2002]

- $$\boxed{F(x_{[1:n]}) = \hat{t}_{[1:n]} = \arg\max_{t_{[1:n]}} \text{score}(x_{[1:n]}, t_{[1:n]})}$$

Learning a global ranker: objective function

[Collins, 2002]

- $$\boxed{F(x_{[1:n]}) = \hat{t}_{[1:n]} = \arg\max_{t_{[1:n]}} \text{score}(x_{[1:n]}, t_{[1:n]})}$$

- $\text{score}(x_{[1:n]}, t_{[1:n]}) = \sum_{r \in R(\mathbf{x},\mathbf{t})} \text{score}(r) =$

  $\sum_{i=1}^{n} \text{score}(r_i) = \sum_{i=1}^{n} \mathbf{w} \cdot \phi(r_i)$

  $\sum_{i=1}^{n} \sum_{j=1}^{d} w_j \cdot \phi_j(r_i) = \sum_{j=1}^{d} \sum_{i=1}^{n} w_j \cdot \phi_j(r_i) =$

  $\sum_{j=1}^{d} w_j \cdot \Phi_j(x_{[1:n]}, t_{[1:n]}) = \mathbf{w} \cdot \Phi(x_{[1:n]}, t_{[1:n]})$

- $\mathbf{w} = (w_1, \ldots, w_d)$ is the weight vector

Learning a global ranker: Perceptron training [Collins, 2002]

Training set: $S = \{(x^i_{[1:n_i]}, t^i_{[1:n_i]})\}_{i=1}^n$

Initialization: set parameters $\mathbf{w} = \mathbf{0}$

repeat for N epochs

for $i = 1 \ldots n$

Use the Viterbi algorithm to compute:

$\hat{t}_{[1:n_i]} = \arg\max_{t_{[1:n_i]}} \mathbf{w} \cdot \Phi(x^i_{[1:n_i]}, t_{[1:n_i]})$

This search makes use of the decomposition of the problem: $r_i, \phi()$

if $(\hat{t}_{[1:n_i]} \neq t^i_{[1:n_i]})$ **then**

$\mathbf{w} = \mathbf{w} + \Phi(x^i_{[1:n_i]}, t^i_{[1:n_i]}) - \Phi(x^i_{[1:n_i]}, \hat{t}_{[1:n_i]})$

end-for

end-repeat

output: $\mathbf{w}$

Learning a global ranker: Perceptron training

[Collins, 2002]

- Use of the voted perceptron with "averaged" parameters
- Application to POS tagging and NP chunking
- Comparison to a MaxEnt-based tagger (MEMM with exactly the same features)
- Simple approach with 11.9% and 5.1% relative error reduction, respectively
- Best paper award at EMNLP-2002

Course Overview

- ...
- **Statistical Learning for Structured NLP**
  - ⋆ Generative models
  - ⋆ The Learning and Inference paradigm
  - ⋆ Re-ranking candidate solutions
  - ⋆ Structure learning with global linear models
    - ∗ Perceptron Global Learning
    - ∗ Max-margin learning algorithms
    - ∗ Conditional Random Fields
- ...

Max-margin Training Algorithms (I)

[Bartlett et al., 2004]

- Same algorithm used for ranking. It generalizes well to work with all solutions when the problem decomposes naturally in parts (labeled sequences, parsing trees, etc.)
- Iterative algorithm with EG (multiplicative) updates
- Works on the setting of **Max-Margin Markov Networks** **[Taskar et al., 2003]**
- Presented as a batch learning algorithm but the online version can also be derived
- See Michael Collins' slides from CoNLL-2006 (PDF file)

Max-margin Training Algorithms (II)

[Tsochantaridis et al., 2004]

- Works on the alternative setting of **Hidden Markov Support Vector Machines** **[Altun et al., 2003]**

- Implemented and publicly available with several instantiations at the SVMstruct site:

  `www.cs.cornell.edu/People/tj/svm_light/svm_struct.html`

Max-margin Training Algorithms (II)

[Tsochantaridis et al., 2004]

- Structured output space $\mathcal{Y}$ (e.g., parsing trees)
- Discriminant function $F : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathbb{R}$
- Objective function: $f(x; \mathbf{w}) = \arg\max_{y \in \mathcal{Y}} F(x, y; \mathbf{w})$
- F restricted to be a linear function: $F(x, y; \mathbf{w}) = \Phi(x, y) \cdot \mathbf{w}$
- For instance, if the problem is parsing (grammar learning), with the PCFG paradigm in mind, $F(x, y; \mathbf{w})$ gives the score of the y parse as the sum of scores of the productions involved in the derivation of the parse tree.

Max-margin Training Algorithms (II)

[Tsochantaridis et al., 2004]

- This can be done by setting the representation function $\Phi(x, y)$ to return a histogram vector with one position for each production rule and its value as the number of times applied in y .
- $x =$"the dog chased the cat"
- $y =$

```
            ___ S ____
           /          \
         NP            VP
         /\           /  \
        /  \         /    NP
       /    \       /     / \
     Det     N     V   Det   N
      |      |     |    |    |
     the    dog chased the   cat
```

Max-margin Training Algorithms (II)

[Tsochantaridis et al., 2004]

rule	$\mathbf{w}_{PCFG}$	$\Phi(x,y)$	$\mathbf{w}_{DL}$
$S \longrightarrow NP\ VP$	0.54	1	3.45
$S \longrightarrow NP$	0.01	0	-0.01
$NP \longrightarrow Det\ N$	0.37	2	8.70
$VP \longrightarrow V\ NP$	0.45	1	2.27
$\ldots$	$\ldots$	$\ldots$	$\ldots$
$Det \longrightarrow the$	0.79	2	3.12
$N \longrightarrow dog$	0.04	1	0.23
$N \longrightarrow cat$	0.03	1	0.25
$N \longrightarrow mouse$	0.06	0	0.34
$V \longrightarrow chased$	0.01	1	0.18
$\ldots$	$\ldots$	$\ldots$	$\ldots$

$$F(x,y;\mathbf{w}_{DL}) = \mathbf{w}_{DL} \cdot \Phi(x,y) = 1 * 3.45 + 2 * 8.70 + \cdots + 1 * 0.18 = 30.02$$

Max-margin Training Algorithms (II)

[Tsochantaridis et al., 2004]

- **Goal** of the learning: acquiring the weight vector $\mathbf{w}_{DL}$ so that performance of $f(x;\mathbf{w}_{DL})$ is maximized on the training set.

- New definition of *margin*: difference between score given to the pair (x_i, y_i) and the best scored (x_i, y), for $y \in \mathcal{Y}, y \neq y_i$

- Standard SVM (soft-margin) learning can be stated:

  $\boxed{\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i}$, subject to $\forall i\, \xi_i \geq 0$ and

  $$\forall i\, \forall y \neq y_i \in \mathcal{Y} \,:\, \mathbf{w} \cdot \Phi(x_i, y_i) - \mathbf{w} \cdot \Phi(x_i, y) \geq 1 - \xi_i$$

  ...but, there is an exponential number of constraints:

Max-margin Training Algorithms (II)

[Tsochantaridis et al., 2004]

- Loss functions other than 0/1-loss:
  - ⋆ Scaling slack variables with the inverse loss (= multiplying the violation by the loss). New form of constraints:

    $$\forall i \, \forall y \neq y_i \in \mathcal{Y} : \mathbf{w} \cdot \Phi(x_i, y_i) - \mathbf{w} \cdot \Phi(x_i, y) \geq 1 - \frac{\xi_i}{L(x_i, y_i, y)}$$

  - ⋆ Re-scale the margin.

    $$\forall i \, \forall y \neq y_i \in \mathcal{Y} : \mathbf{w} \cdot \Phi(x_i, y_i) - \mathbf{w} \cdot \Phi(x_i, y) \geq L(x_i, y_i, y) - \xi_i$$

Max-margin Training Algorithms (II)

[Tsochantaridis et al., 2004]

Input: $S = \{(x_i, y_i)\}_{i=1}^n, C, \epsilon$

Initialization: $S_i \leftarrow \emptyset$, for all $i = 1, \ldots, n$
repeat
for $i = 1, \ldots, n$ **do**

Set up cost function:
$$H(y) = (1 - (\mathbf{w} \cdot \Phi(x_i, y_i) - \mathbf{w} \cdot \Phi(x_i, y))) * L(x_i, y_i, y)$$
where $\mathbf{w} = \sum_j \sum_{y \in S_j} \alpha_{j,y}(\Phi(x_j, y_j) - \Phi(x_j, y))$

compute $\hat{y} = \arg\max_{y \in \mathcal{Y}} H(y)$
$\xi_i = \max\{0, \max_{y \in S_i} H(y)\}$

if $H(\hat{y}) > \xi_i + \epsilon$ **then**
$S_i \leftarrow S_i \cup \{\hat{y}\}$
$\alpha_S \leftarrow$ optimize dual over $S, S = \cup_i S_i$ **end-if**
end-for
until no S_i has changed during iteration

Max-margin Training Algorithms (II)

[Tsochantaridis et al., 2004]

- The previous on-line training algorithm converges in polynomial time. There always exist a polinomially-sized subset of constraints so that the corresponding solution fulfills the full set of constraints with a precision of ϵ.

- An instantiation of the algorithm for a concrete problem must define: $\Phi(x, y)$, $L(x, y, \hat{y})$, and the "$\arg\max$" search in the calculation of $\hat{y}$.

- At the $\mathrm{SVMstruct}$ website one can find several instantiations of the algorithm, including inference algorithms for HMM-like sequential tagging and PCFG-based parsing.

Course Overview

- ...
- **Statistical Learning for Structured NLP**
  - ⋆ Generative models
  - ⋆ The Learning and Inference paradigm
  - ⋆ Re-ranking candidate solutions
  - ⋆ Structure learning with global linear models
    - ∗ Perceptron Global Learning
    - ∗ Max-margin learning algorithms
    - ∗ Conditional Random Fields
- ...

Probabilistic Global Models: CRFs

Motivation

- Problems of MEMMs and variants of chained sequential inference schemes with local classifiers **[Punyakanok et al., 2002; Giménez and Màrquez, 2003; Kudo and Matsumoto, 2001]**
  - ⋆ Training is local, without taking into account loss functions derived from global performance measures
  - ⋆ *Label bias problem*

Probabilistic Global Models: CRFs

Motivation

- Problems of MEMMs and variants of chained sequential inference schemes with local classifiers **[Punyakanok et al., 2002; Giménez and Màrquez, 2003; Kudo and Matsumoto, 2001]**
  - ⋆ Training is local, without taking into account loss functions derived from global performance measures
  - ⋆ *Label bias problem*
- The problems of generative models are also well-known (e.g., they cannot use arbitrary representations on the inputs)

Probabilistic Global Models: CRFs

Motivation

- Problems of MEMMs and variants of chained sequential inference schemes with local classifiers **[Punyakanok et al., 2002; Giménez and Màrquez, 2003; Kudo and Matsumoto, 2001]**
  - ⋆ Training is local, without taking into account loss functions derived from global performance measures
  - ⋆ *Label bias problem*
- The problems of generative models are also well-known (e.g., they cannot use arbitrary representations on the inputs)
- Conditional Random Fields **[Lafferty, McCallum, and Pereira 2001]** try to get the best of both worlds without any of the shortcomings

Conditional Random Fields

- CRF is a conditional model $p(\mathbf{y}|\mathbf{x})$
- It defines a single log-linear distribution over label structure ($\mathbf{y}$) given the observations ($\mathbf{x}$)
- CRF can be viewed as an undirected graphical model or Markov random field globally conditioned on $\mathbf{x}$

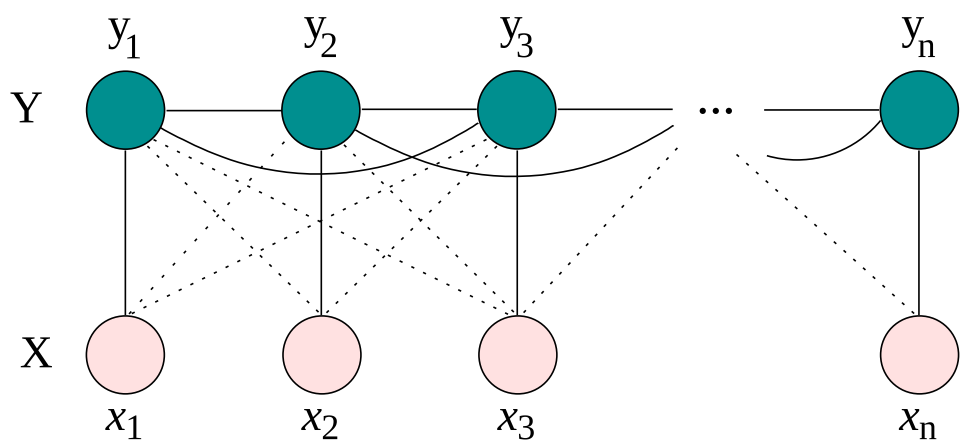
Conditional Random Fields

- CRF is a conditional model $p(\mathbf{y}|\mathbf{x})$
- It defines a single log-linear distribution over label structure ($\mathbf{y}$) given the observations ($\mathbf{x}$)
- CRF can be viewed as an undirected graphical model or Markov random field globally conditioned on $\mathbf{x}$
- A graphical model is a family of probability distributions that factorize according to an underlying graph.
- Represent the distribution over a large number of random variables by a product of local functions that each depend only on a small number of variables

Conditional Random Fields

- The most common instantiation is the Linear-chain CRF model

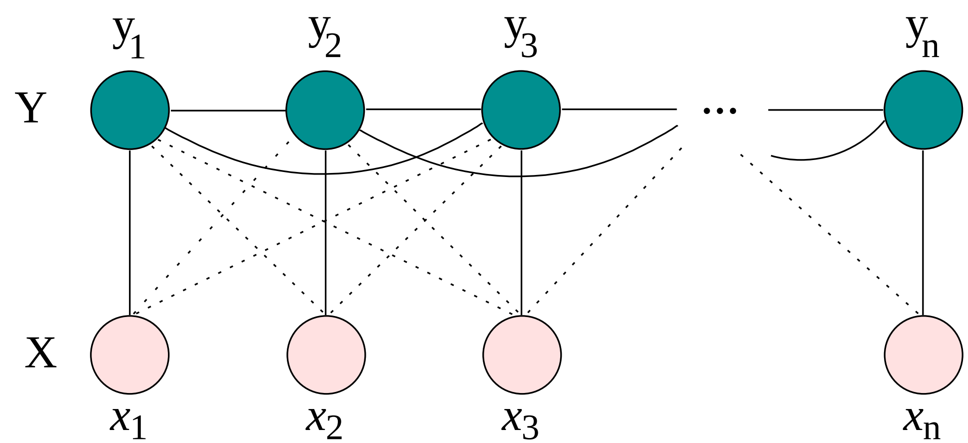
Graphical Model of a linear chain CRF



Conditional Random Fields

- The most common instantiation is the Linear-chain CRF model

Graphical Model of a linear chain CRF



- Two types of dependencies: (y_{i-1}, y_i) and $(\mathbf{x}, y_i)$
- Training and decoding are efficient
- Direct application to all (NLP) sequential labeling problems

Linear-chain CRFs

- $p(\mathbf{y}|\mathbf{x})$ factorize in a normalized product of potential functions of the form: $\boxed{\exp(\sum_j \lambda_j t_j(y_{i-1}, y_i, \mathbf{x}, i) + \sum_k \mu_k s_k(y_i, \mathbf{x}, i))}$
- $t_j(y_{i-1}, y_i, \mathbf{x}, i)$ is a transition feature function
- $s_k(y_i, \mathbf{x}, i))$ is a state feature function
- λ_j and μ_k are the parameters to be estimated from training data

Linear-chain CRFs

- $p(\mathbf{y}|\mathbf{x})$ factorize in a normalized product of potential functions of the form: $\boxed{\exp(\sum_j \lambda_j t_j(y_{i-1}, y_i, \mathbf{x}, i) + \sum_k \mu_k s_k(y_i, \mathbf{x}, i))}$
- $t_j(y_{i-1}, y_i, \mathbf{x}, i)$ is a transition feature function
- $s_k(y_i, \mathbf{x}, i))$ is a state feature function
- λ_j and μ_k are the parameters to be estimated from training data

t_j and s_k are indicator functions. Example:

$$t_j(y_{i-1}, y_i, \mathbf{x}, i) = \begin{cases} 1 & \text{if } y_{i-1} = \text{IN and } y_i = \text{NNP and } x_i = \text{“September”} \\ 0 & \text{otherwise} \end{cases}$$

Linear-chain CRFs

- Expressing t_j and s_k as a general $f_j(y_{i-1}, y_i, \mathbf{x}, i)$
- ...and considering $F_j(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, \mathbf{x}, i)$
- The probability of a label sequence **y** given an observation sequence **x** is

Linear-chain CRFs

- Expressing t_j and s_k as a general $f_j(y_{i-1}, y_i, \mathbf{x}, i)$
- ...and considering $F_j(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, \mathbf{x}, i)$
- The probability of a label sequence $\mathbf{y}$ given an observation sequence $\mathbf{x}$ is

$$\boxed{p(\mathbf{y}|\mathbf{x}, \lambda) = \frac{1}{Z(\mathbf{x})} \exp(\sum_j \lambda_j F_j(\mathbf{y}, \mathbf{x}))}$$

- where $Z(\mathbf{x})$ is a normalization factor

Linear-chain CRFs

- Expressing t_j and s_k as a general $f_j(y_{i-1}, y_i, \mathbf{x}, i)$
- ...and considering $F_j(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, \mathbf{x}, i)$
- The probability of a label sequence **y** given an observation sequence **x** is

$$\boxed{p(\mathbf{y}|\mathbf{x}, \lambda) = \frac{1}{Z(\mathbf{x})} \exp(\sum_j \lambda_j F_j(\mathbf{y}, \mathbf{x}))}$$

- where $Z(\mathbf{x})$ is a normalization factor
- This is a log-linear probability distribution similar to ME

CRFs: Parameter estimation

- Optimize the *conditional log-likelihood* of λ on the training set
- $l(\lambda) = \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}|\mathbf{y}^{(i)})$
- $l(\lambda) = \sum_{i=1}^{N} \sum_j \lambda_j F_j(\mathbf{y}^{(i)}, \mathbf{x}^{(i)})) - \sum_{i=1}^{N} \log Z(\mathbf{x}^{(i)}) - \sum_j \frac{\lambda_j^2}{2\sigma^2}$
- $\frac{1}{2\sigma^2}$ is a regularization parameter
- Several methods can be used for training
- Cost: $O(nM^2NG)$

CRFs: inference

- Decoding: $\boxed{\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}, \lambda)}$
- This can be done by using variants of the Viterbi dynamic programming for HMMs

CRFs: applications

- NLP: Text classification, POS tagging, chunking, named-entity recognition, semantic role labeling, etc.
  (See the survey by **[Sutton and McCallum, 2006]**)
- Bioinformatics
- Computer vision

Course Overview

- ...
- **Statistical Learning for Structured NLP**
  - ⋆ Generative models
  - ⋆ The Learning and Inference paradigm
  - ⋆ Re-ranking candidate solutions
  - ⋆ Structure learning with global linear models
  - ⋆ Conclusions
- ...

Conclusions

- We have reviewed most of the statistical learning techniques for structured Natural Language Processing
  - ⋆ Generative Models
  - ⋆ Learning and Inference paradigm
    - ∗ Examples of using classifiers in chained decisions for sequential and hierarchical annotation: HMM-with-classifiers, PMMs, MEMMs, etc.)
    - ∗ Inference as constraint satisfaction
    - ∗ The FR-Perceptron algorithm
  - ⋆ Reranking global candidates
  - ⋆ Discriminative global learning (MMMN, HMSVM, CRFs, etc.)

Conclusions

Opportunities

- Exploit linguistically rich features and complex dependencies
- Surpass the traditional NLP architecture of a pipeline of processors
- Approach multitask learning
- Design intermediate structures to learn, which are optimal for the global perfomance

Conclusions

Current shortcomings

- No single approach is best in all aspects:
  - ⋆ Generative models can be a good and very efficient alternative in simple problems, but they are not extensible (w.r.t topology and features)
  - ⋆ MEMMs and other inference schemes with local classifiers can work with arbitrary features but they suffer from locality problems (label bias problem). Also, dynamic inference is limiting the type of features on the output.

Conclusions

Current shortcomings

- No single approach is best in all aspects:
  - ⋆ Reranking is efficient and good for complex feature design (on the complete output candidates), but it only provides slight improvements of the performance of a base linguistic processor for the task
  - ⋆ Global models (MMMN, HMSVM, CRFs, etc.) overcome locality problems and obtain better results, but they are computationally very expensive. Again, the dynamic inference limits the type of features on the output

Course Overview

- Introduction and Motivation
  - ⋆ NLP tasks and Learning
  - ⋆ NLP tools in the Web
- Machine Learning for supervised classification
  - ⋆ Machine learning algorithms
  - ⋆ Applications
- Statistical Learning for Structured NLP
- **A case study: Semantic Role Labeling**

Semantic Role Labeling (SRL)

See the accompanying SRL tutorial from ACL-2009:

- Semantic Role Labeling: Past, Present and Future

  `www.lsi.upc.edu/~lluism/tmp/SRL-tutorial-ACL-IJCNLP-2009.pdf`

  `www.lsi.upc.edu/~lluism/tmp/SRL-tutorial-ACL-IJCNLP-2009-references.pdf`

Course Overview

- Introduction and Motivation
- Machine Learning for supervised classification
- Statistical Learning for Structured NLP
- A case study: Semantic Role Labeling
- **Conclusions**

Conclusions

- ...1
- ...
- ...

Conclusions

- ...2
- ...
- ...

Conclusions

- ...3
- ...
- ...

Conclusions

- ...4
- ...
- ...

Ackowledgements

Special thanks to

- Xavier Carreras, Mike Collins, Scott Wen-Tau Yih, Kristina Toutanova and Dan Roth for making available their slides

- IXA and UPV/EHU master organizers for their kind invitation

- Research Projects and institutions funding the research of the GPLN at UPC: MEANING, CHIL, TRANGRAM, ALIADO, TC-STAR, CESS-ECE, OpenMT, FAUST, MOLTO, etc.

- All the attendants to the course!

Thank you very much for your attention!