Overview of NLP Tasks and Featurization

Frank Ferraro – ferraro@umbc.edu
CMSC 473/673

Some slides courtesy & adapted from Jason Eisner

Today's Learning Goals

- Define featurization and some examples
- Define some "classification" terminology
- Learn about NLP Tasks at a high-level, e.g.,
  - Document classification
  - Part of speech tagging
  - Syntactic parsing
  - Entity id/coreference

Helpful ML Terminology Recap (1)

- Model: the (computable) way you're going from inputs/representations of input to labels or scores

- Weights/parameters: collections of vectors that control how the model produces labels/scores from inputs. These are learned.

Helpful ML Terminology Recap (2)

- Model: the (computable) way you're going from inputs/representations of input to labels or scores
- Weights/parameters: collections of vectors that control how the model produces labels/scores from inputs. These are learned.
- Objective function: a function, whose variables are the weights of the model, that we numerically optimize in order to learn appropriate weights based on the labels/scores. The model's weights **are** adjusted.
- Evaluation function: a function that scores how correct the model's predicted labels are. The model's weights **are not** adjusted.
  - The evaluation and objective functions are (likely) different!

Helpful ML Terminology Recap (3)

Learning:

the process of adjusting the model's weights to learn to make good predictions.

Inference / Prediction / Decoding / Classification:

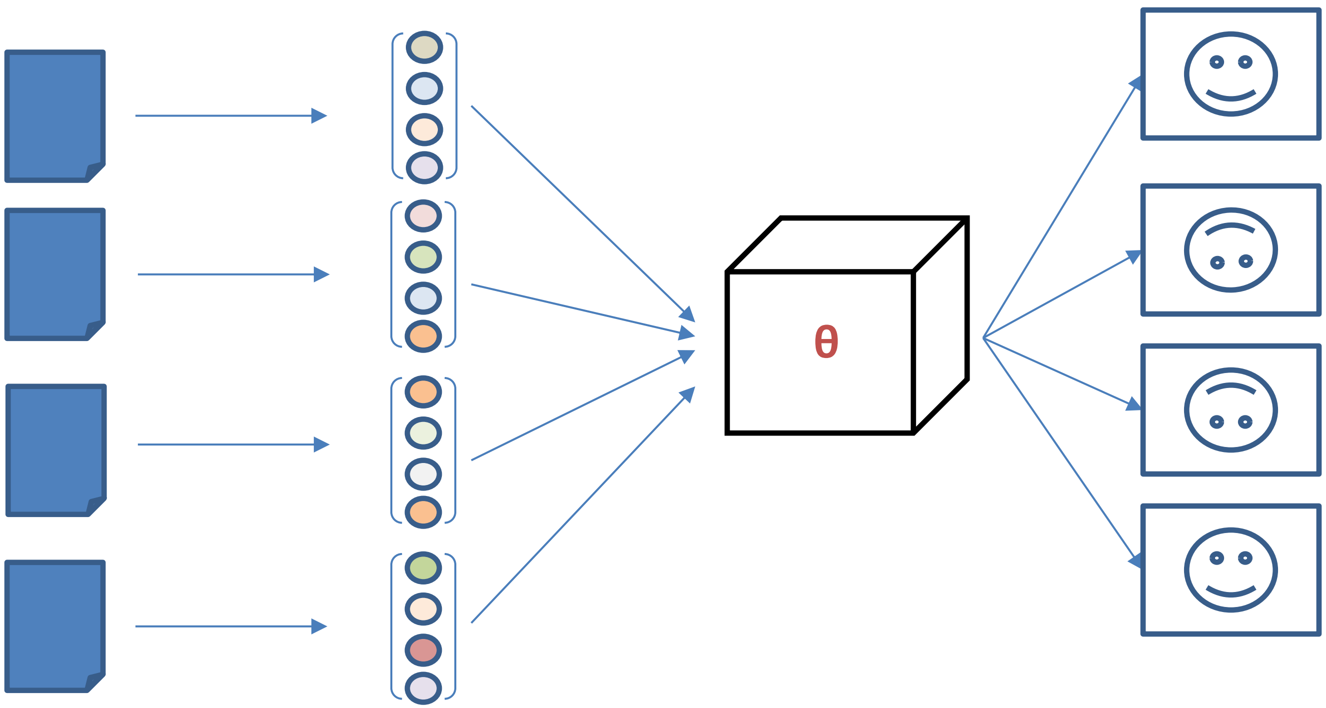
the process of using a model's existing weights to make (hopefully!) good predictions

ML/NLP Framework

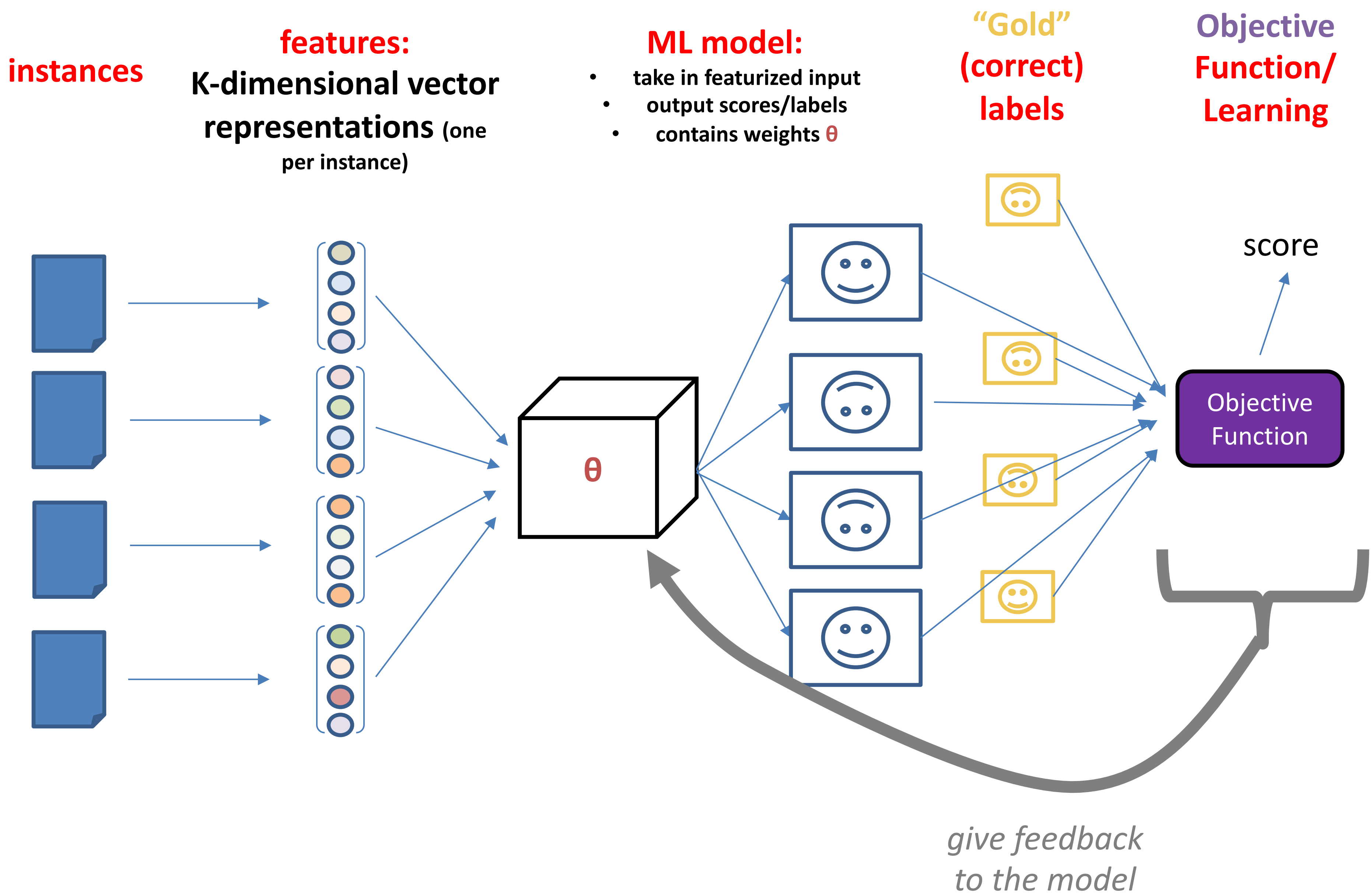
instances

features:
K-dimensional vector representations (one per instance)

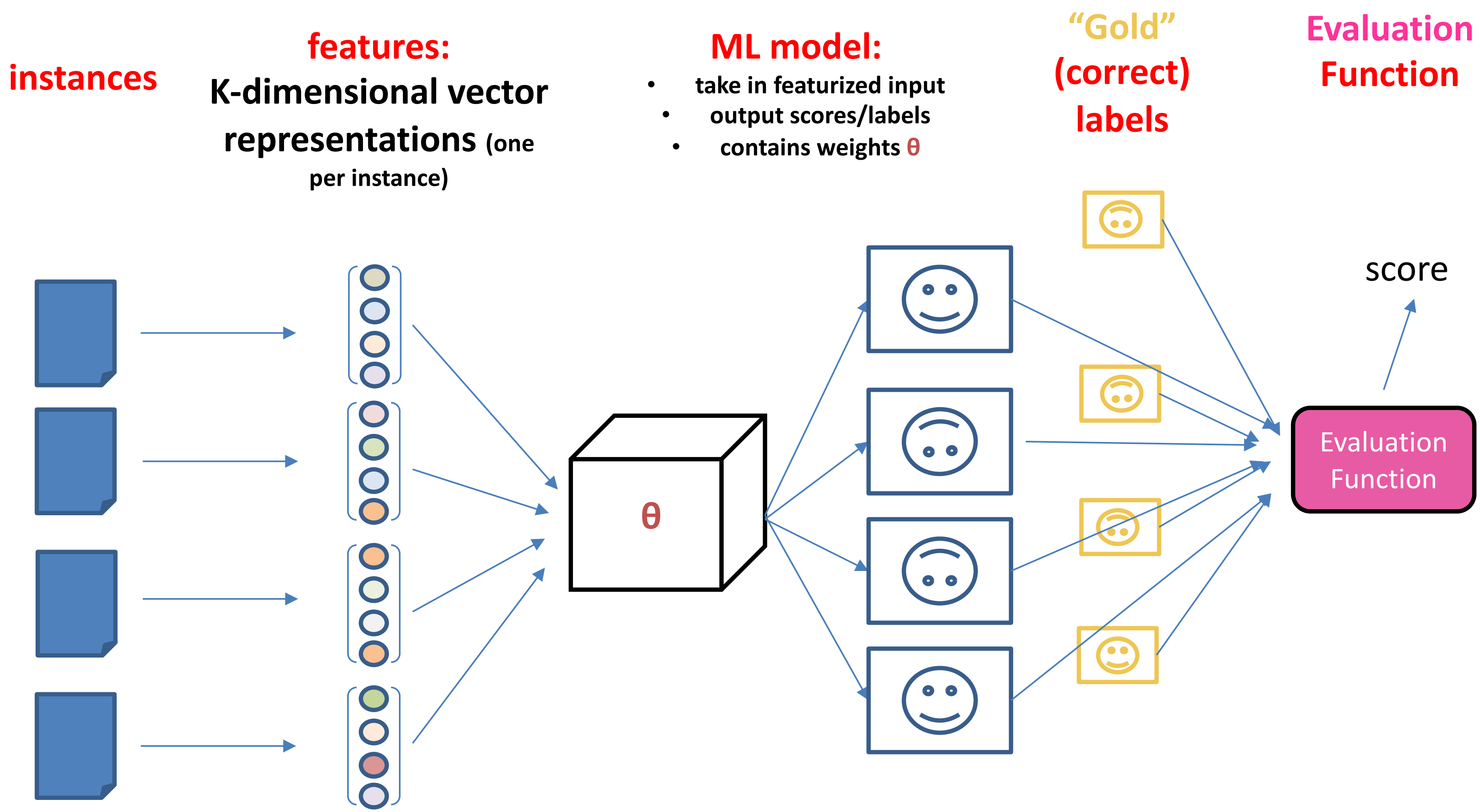
ML model:
- take in featurized input
- output scores/labels
- contains weights θ



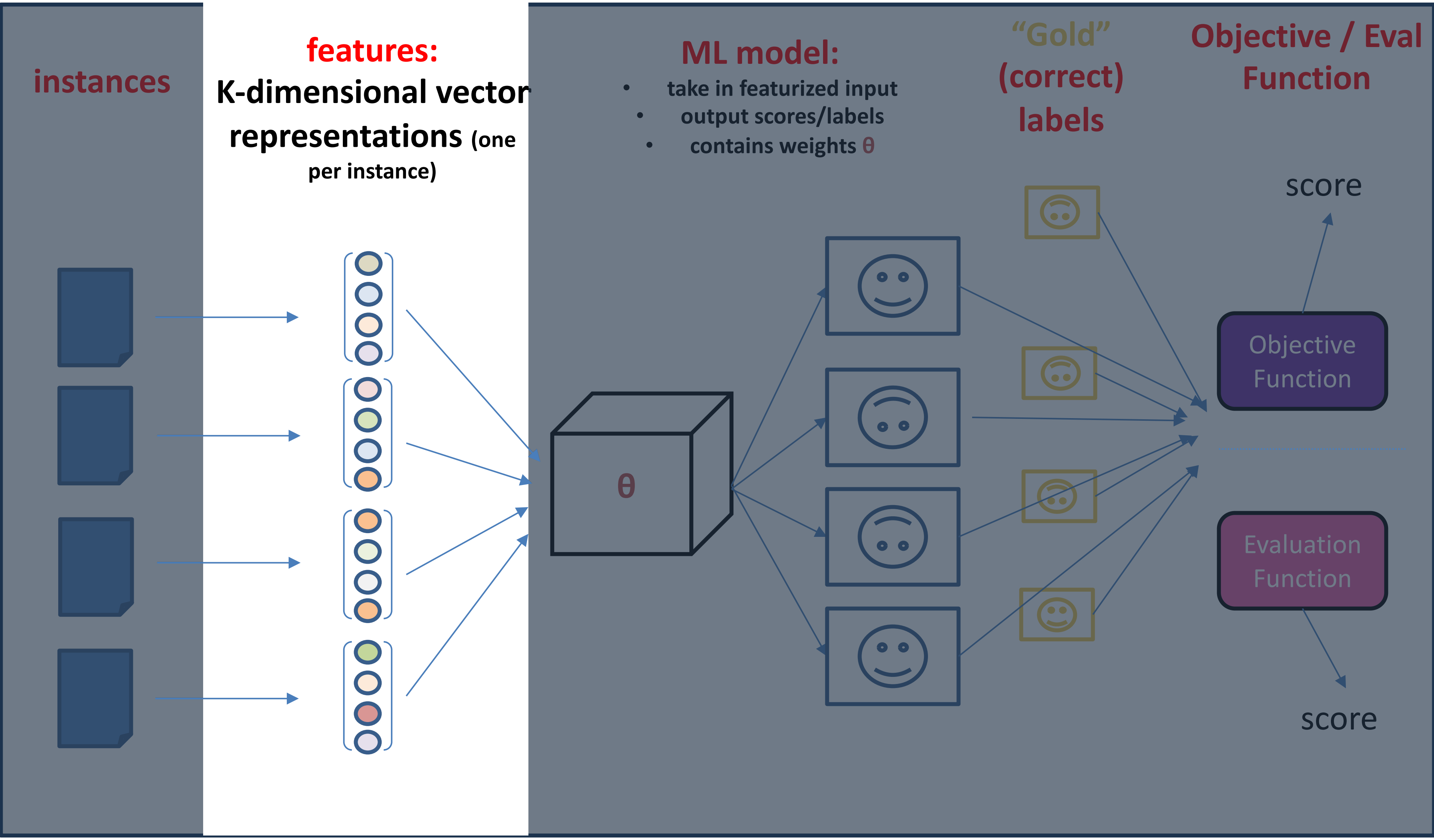
ML/NLP Framework for Learning



ML/NLP Framework for Prediction



First: Featurization / Encoding / Representation



ML Term: "Featurization"

The procedure of extracting **features** for some input

Often viewed as a K-dimensional vector function f of the input language x

$$f(x) = (\underbrace{f_1(x)}, \dots, \underbrace{f_K(x)})$$

Each of these is a feature
(/feature function)

ML Term: “Featurization”

The procedure of extracting **features** for some input

Often viewed as a K-dimensional vector function *f* of the input language *x*

$$f(x) = (f_1(x), \ldots, f_K(x))$$

In supervised settings, it can equivalently be viewed as a K-dimensional vector function *f* of the input language *x* and a potential label *y*

$$f(x, y) = (f_1(x, y), \ldots, f_K(x, y))$$

Features can be thought of as “soft” rules

- E.g., POSITIVE sentiments tweets *may* be more likely to have the word “happy”

Defining Appropriate Features

Feature functions help extract useful features (characteristics) of the data

They turn *data* into *numbers*

Features that are not 0 are said to have **fired**

Defining Appropriate Features

Feature functions help extract useful features (characteristics) of the data

They turn *data* into *numbers*

Features that are not 0 are said to have fired

You can define classes of features by *templating* (we'll come back to this!)

Often binary-valued (0 or 1), but can be real-valued

Three Common Types of Featurization in NLP

1. Bag-of-words (or bag-of-characters, bag-of-relations)
2. Linguistically-inspired features
3. Dense features via embeddings

Three Common Types of Featurization in NLP

1. Bag-of-words (or bag-of-characters, bag-of-relations)
    - easy to define / extract
    - sometimes still very useful
2. Linguistically-inspired features
3. Dense features via embeddings

Three Common Types of Featurization in NLP

1. Bag-of-words (or bag-of-characters, bag-of-relations)
   - easy to define / extract
   - sometimes still very useful

2. Linguistically-inspired features
   - harder to define
   - helpful for interpretation
   - depending on task: conceptually helpful
   - currently, not freq. used

3. Dense features via embeddings

Three Common Types of Featurization in NLP

1. Bag-of-words (or bag-of-characters, bag-of-relations)
   - easy to define / extract
   - sometimes still very useful
2. Linguistically-inspired features
   - harder to define
   - helpful for interpretation
   - depending on task: conceptually helpful
   - currently, not freq. used
3. Dense features via embeddings
   - harder to define
   - harder to extract (unless there's a model to run)
   - currently: freq. used

Three Common Types of Featurization in NLP

1. Bag-of-words (or bag-of-characters, bag-of-relations)
   – Identify ***unique*** sufficient atomic sub-parts (e.g., words in a document)
   – Define simple features over these, e.g.,
      - Binary (0 or 1) ➔ indicating presence
      - Natural numbers ➔ indicating number of times in a context
      - Real-valued ➔ various other score (we'll see examples throughout the semester)
2. Linguistically-inspired features
3. Dense features via embeddings

Three Common Types of Featurization in NLP

1. Bag-of-words (or bag-of-characters, bag-of-relations)
   – Identify ***unique*** sufficient atomic sub-parts (e.g., words in a document)
   – Define simple features over these, e.g.,
     - Binary (0 or 1) ➔ indicating presence
     - Natural numbers ➔ indicating number of times in a context
     - Real-valued ➔ various other score (we'll see examples throughout the semester)
2. Linguistically-inspired features
   – Define features from words, word spans, or linguistic-based annotations extracted from the document
3. Dense features via embeddings

Three Common Types of Featurization in NLP

1. Bag-of-words (or bag-of-characters, bag-of-relations)
   - Identify ***unique*** sufficient atomic sub-parts (e.g., words in a document)
   - Define simple features over these, e.g.,
     - Binary (0 or 1) ➔ indicating presence
     - Natural numbers ➔ indicating number of times in a context
     - Real-valued ➔ various other score (we'll see examples throughout the semester)
2. Linguistically-inspired features
   - Define features from words, word spans, or linguistic-based annotations extracted from the document
3. Dense features via embeddings
   - Compute/extract a real-valued vector, e.g., from word2vec, ELMO, BERT, …

Example: Document Classification via Bag-of-Words Features

Electronic alerts have been used to assist the authorities in moments of chaos and potential danger: after the Boston bombing in 2013, when the Boston suspects were still at large, and last month in Los Angeles, during an active shooter scare at the airport.

TECH

NOT TECH

Let’s make a core assumption: the **label** can be predicted from **counts of individual word types**

Example: Document Classification via Bag-of-Words Features

Electronic alerts have been used to assist the authorities in moments of chaos and potential danger: after the Boston bombing in 2013, when the Boston suspects were still at large, and last month in Los Angeles, during an active shooter scare at the airport.

feature extraction

TECH

NOT TECH

With V word types, define V feature functions $f_i(x)$ as

$f_i(x) =$# of times word type *i* appears in document x

Core assumption: the label can be predicted from counts of individual word types

Example: Document Classification via Bag-of-Words Features

Electronic alerts have been used to assist the authorities in moments of chaos and potential danger: after the Boston bombing in 2013, when the Boston suspects were still at large, and last month in Los Angeles, during an active shooter scare at the airport.

feature extraction

TECH

NOT TECH

With V word types, define V feature functions $f_i(x)$ as

$f_i(x)$ =# of times word type *i* appears in document x

$$f(x) = \left(f_i(x)\right)_i^V$$

Core assumption: the label can be predicted from counts of individual word types

Example: Document Classification via Bag-of-Words Features

Electronic alerts have been used to assist the authorities in moments of chaos and potential danger: after the Boston bombing in 2013, when the Boston suspects were still at large, and last month in Los Angeles, during an active shooter scare at the airport.

TECH

NOT TECH

feature extraction

feature $f_i(x)$	value
alerts	1
assist	1
bombing	1
Boston	2
…	
sniffle	0
…	

Core assumption: the label can be predicted from counts of individual word types

Example: Document Classification with Bag-of-Words Features

Electronic alerts have been used to assist the authorities in moments of chaos and potential danger: after the Boston bombing in 2013, when the Boston suspects were still at large, and last month in Los Angeles, during an active shooter scare at the airport.

TECH

NOT TECH

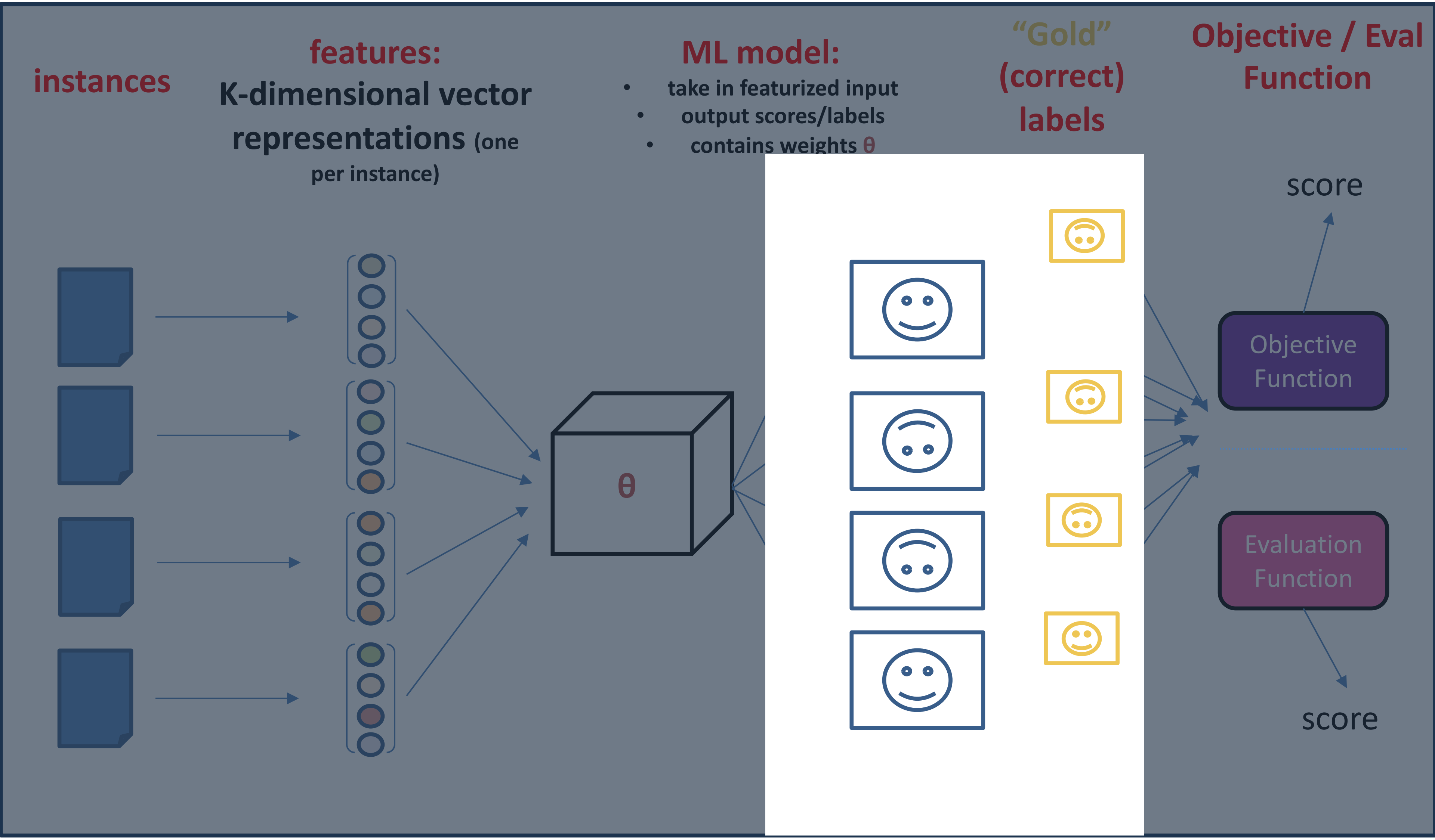
feature	weight
alerts	.043
assist	-0.25
bombing	0.8
Boston	-0.00001
…	

w: weights

feature $f_i(x)$	value
alerts	1
assist	1
bombing	1
Boston	2
…	

f(**x**): "bag of words"

Second: Classification Terminology



Classification Types (Terminology)

Name	Number of Tasks (Domains) Labels are Associated with	# Label Types	Example
(Binary) Classification			
Multi-class Classification			
Multi-label Classification			
Multi-task Classification			

Classification Types (Terminology)

Name	Number of Tasks (Domains) Labels are Associated with	# Label Types	Example
(Binary) Classification	1	2	Sentiment: Choose one of {positive or negative}
Multi-class Classification			
Multi-label Classification			
Multi-task Classification			

Classification Types (Terminology)

Name	Number of Tasks (Domains) Labels are Associated with	# Label Types	Example
(Binary) Classification	1	2	Sentiment: Choose one of {positive or negative}
Multi-class Classification	1	> 2	Part-of-speech: Choose one of {Noun, Verb, Det, Prep, …}
Multi-label Classification			
Multi-task Classification			

Classification Types (Terminology)

Name	Number of Tasks (Domains) Labels are Associated with	# Label Types	Example
(Binary) Classification	1	2	Sentiment: Choose one of {positive or negative}
Multi-class Classification	1	> 2	Part-of-speech: Choose one of {Noun, Verb, Det, Prep, …}
Multi-label Classification	1	> 2	Sentiment: Choose multiple of {positive, angry, sad, excited, …}
Multi-task Classification			

Classification Types (Terminology)

Name	Number of Tasks (Domains) Labels are Associated with	# Label Types	Example
(Binary) Classification	1	2	Sentiment: Choose one of {positive or negative}
Multi-class Classification	1	> 2	Part-of-speech: Choose one of {Noun, Verb, Det, Prep, …}
Multi-label Classification	1	> 2	Sentiment: Choose multiple of {positive, angry, sad, excited, …}
Multi-task Classification	> 1	Per task: 2 or > 2 (can apply to binary or multi-class)	Task 1: part-of-speech Task 2: named entity tagging … ---------------------- Task 1: document labeling Task 2: sentiment

Text Annotation Tasks

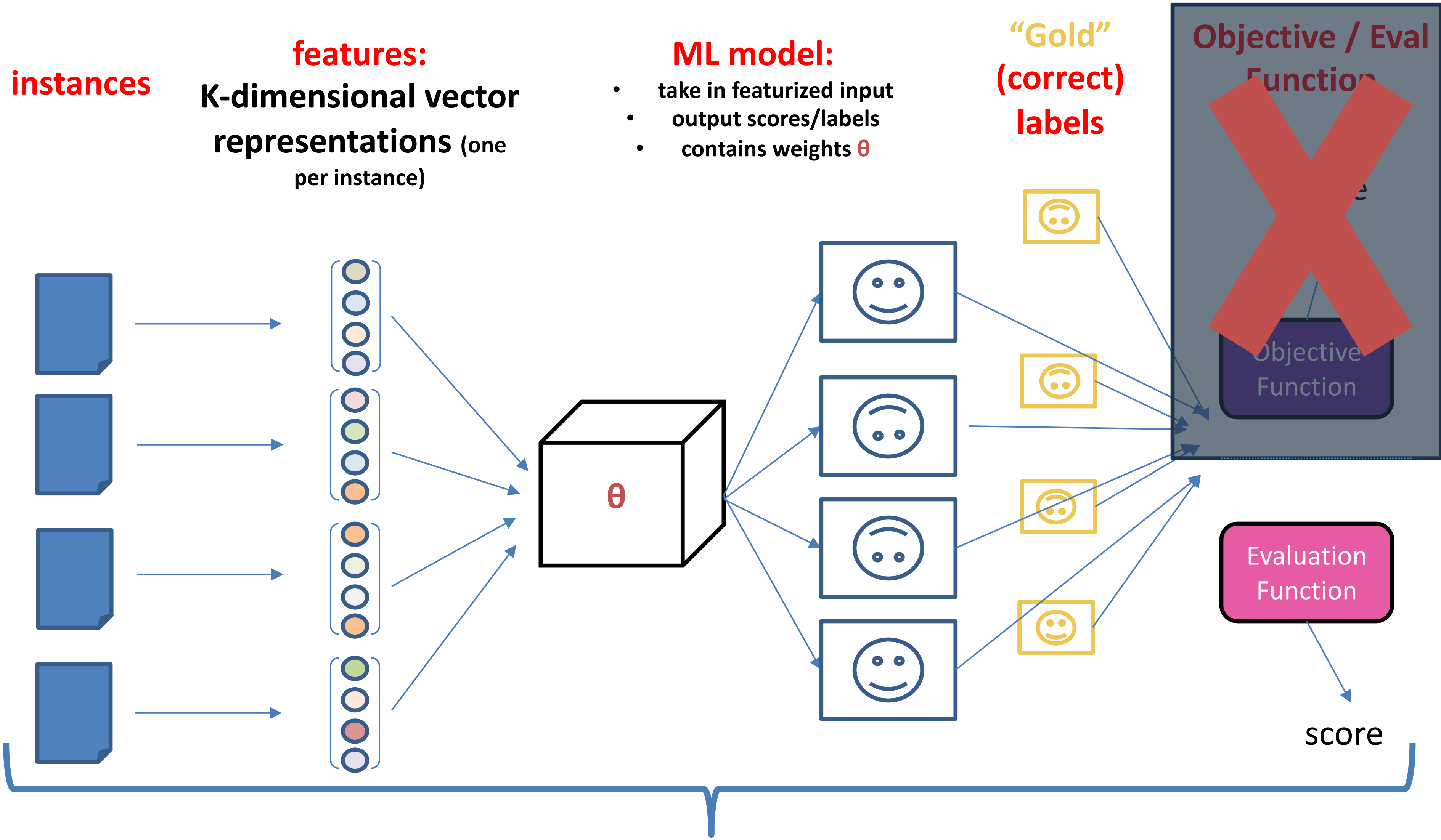
1. Classify the entire document
2. Classify word tokens individually
3. Classify word tokens in a sequence
4. Identify phrases ("chunking")
5. Syntactic annotation (parsing)
6. Semantic annotation
7. Text generation

A demo (and note) about `transformers.pipeline`

`transformers.pipeline` (API, tutorial)

- Many predefined tasks
- Allows for easy-to-use inference (prediction)

`transformers.pipeline` and Inference



`transformers.pipeline` make the inference portion much easier… sometimes

A demo (and note) about `transformers.pipeline`

`transformers.pipeline` (API, tutorial)

- Many predefined tasks
- Allows for easy-to-use inference (prediction)

But…

- what if your task isn't there?
- how do you decide what model to use?! (nearly 3k models in https://huggingface.co/models)
- what if you want to use another model?

Tasks ↔ `pipeline`

"Tasks" in this deck	≅ pipeline task= (not exhaustive)
Classify the entire document	text-classification (if there's a model w/ your labels)
Classify word tokens individually	text-classification
Classify word tokens in a sequence	token-classification
Identify phrases ("chunking")	token-classification
Syntactic annotation (parsing)	N/A, or token-classification or text-generation
Semantic annotation	N/A, or token-classification or text-generation
Text generation	• question-answering • translation • text-generation

Text Annotation Tasks

1. Classify the entire document ("text categorization")
2. Classify individual word tokens
3. Classify word tokens in a sequence
4. Identify phrases ("chunking")
5. Syntactic annotation (parsing)
6. Semantic annotation

Slide courtesy Jason Eisner, with mild edits

Text Classification

Assigning subject categories, topics, or genres

Spam detection

Authorship identification

Age/gender identification

Language Identification

Sentiment analysis

…

Text Classification

Assigning subject categories, topics, or genres

Spam detection

Authorship identification

Age/gender identification

Language Identification

Sentiment analysis

…

Input:

a document

a fixed set of classes $C = \{c_1, c_2,\ldots, c_J\}$

Output: a predicted class c from C

Text Classification

Assigning subject categories, topics, or genres

Spam detection

Authorship identification

Age/gender identification

Language Identification

Sentiment analysis

…

Input:

a ~~document~~ linguistic blob

a fixed set of classes $C = \{c_1, c_2,\ldots, c_J\}$

Output: a predicted class c from C

Text Classification: Hand-coded Rules?

Assigning subject categories, topics, or genres

Spam detection

Authorship identification

Age/gender identification

Language Identification

Sentiment analysis

…

Rules based on combinations of words or other features

spam: black-list-address OR (“dollars” AND “have been selected”)

Accuracy can be high

If rules carefully refined by expert

Building and maintaining these rules is expensive

Can humans faithfully assign uncertainty?

Text Classification: Supervised Machine Learning

Assigning subject categories, topics, or genres

Spam detection

Authorship identification

Age/gender identification

Language Identification

Sentiment analysis

…

Input:

a document d

a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

A training set of m hand-labeled documents $(d_1,c_1),\ldots,(d_m,c_m)$

Output:

a learned classifier γ that maps documents to classes

Text Classification: Supervised Machine Learning

Assigning subject categories, topics, or genres

Spam detection

Authorship identification

Age/gender identification

Language Identification

Sentiment analysis

…

Input:

a document d

a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

A training set of m hand-labeled documents $(d_1,c_1),\ldots,(d_m,c_m)$

Output:

a learned classifier γ that maps documents to classes

Naïve Bayes

Logistic regression

Neural network

Support-vector machines

k-Nearest Neighbors

…

Text Annotation Tasks

1. Classify the entire document ("text categorization")
2. Classify individual word tokens
3. Classify word tokens in a sequence
4. Identify phrases ("chunking")
5. Syntactic annotation (parsing)
6. Semantic annotation
7. Text generation

p(class | token in context)

Word Sense Disambiguation (WSD)

Problem:

The company said the *plant* is still operating ...

⇒ (A) Manufacturing plant or

⇒ (B) Living plant

Training Data: Build a special classifier just for tokens of "plant"

Sense	Context
(1) Manufacturing	... union responses to *plant* closures . ...
" "	... computer disk drive *plant* located in ...
" "	company manufacturing *plant* is in Orlando ...
(2) Living	... animal rather than *plant* tissues can be ...
" "	... to strain microscopic *plant* life from the ...
" "	and Golgi apparatus of *plant* and animal cells

Test Data:

Sense	Context
???	... vinyl chloride monomer *plant* , which is ...
???	... molecules found in *plant* tissue from the ...

Slide courtesy Jason Eisner, with mild edits

slide courtesy of D. Yarowsky

p(class | token in context)

WSD for Machine Translation

(English → Spanish)

Problem:

... He wrote the last **sentence** two years later ...

⇒ *sentencia* (legal sentence) or

⇒ *frase* (grammatical sentence)

Training Data:

Build a special classifier just for tokens of "sentence"

Translation	Context
(1) sentencia	... for a maximum *sentence* for a young offender ...
" "	... of the minimum *sentence* of seven years in jail ...
" "	... were under the *sentence* of death at that time ...
(2) frase	... read the second *sentence* because it is just as ...
" "	... The next *sentence* is a very important ...
" "	... It is the second *sentence* which I think is at ...

Test Data:

Translation	Context
???	... cannot criticize a *sentence* handed down by ...
???	... listen to this *sentence* uttered by a former ...

p(class | token in context)

Accent Restoration in Spanish & French

Problem:

Input: ... deja travaille cote a cote ...

$\Downarrow$

Output: ... déjà travaillé côte à côte ...

Examples:

... appeler l'autre **cote** de l'atlantique ...

$\Rightarrow$ *côté* (meaning side) or

$\Rightarrow$ *côte* (meaning coast)

... une famille des **pecheurs** ...

$\Rightarrow$ *pêcheurs* (meaning fishermen) or

$\Rightarrow$ *pécheurs* (meaning sinners)

p(class | token in context)

Accent Restoration in Spanish & French

Training Data:

Pattern	Context
(1) côté	... du laisser de *cote* faute de temps ...
" "	... appeler l' autre *cote* de l' atlantique ...
" "	... passe de notre *cote* de la frontiere ...
(2) côte	... vivre sur notre *cote* ouest toujours ...
" "	... creer sur la *cote* du labrador des ...
" "	travaillaient cote a *cote* , ils avaient ...

Test Data:

Pattern	Context
???	... passe de notre *cote* de la frontiere ...
???	... creer sur la *cote* du labrador des ...

p(class | token in context)

Capitalization Restoration

Problem:

… FRIED CHICKEN, **TURKEY** SANDWICHES AND FROZEN ...

⇒ *turkey* (the *bird*) or

⇒ *Turkey* (the *country*)

Training Data:

Capitalization	Context
(1) turkey	… OF FRIED CHICKEN , TURKEY SANDWICHES AND FROZEN …
” ”	… NTS A POUND , WHILE TURKEY PRICES ROSE 1.2 CENTS …
” ”	… PLAY , REAL GRADE-A TURKEY , WHICH ONLY A PRICE …
(2) Turkey	… INUNDATED EASTERN TURKEY AFTER THE EARLIER …
” ”	… FEELINGS TOWARD TURKEY SURFACED WHEN GREECE …
” ”	… THE CONTRACT WITH TURKEY WILL PROVIDE OPPORTU…

Test Data:

Capitalization	Context
???	… NECK LIKE THAT OF A TURKEY ON A CHOPPING BLOCK …
???	… PROBLEM IS THAT TURKEY IS NOT A EUROPEAN …

p(class | token in context)

Text-to-Speech Synthesis

Problem:

... slightly elevated *lead* levels ...

⇒ *lɛd* (as in *lead mine*) or

⇒ *li:d* (as in *lead role*)

Training Data:

Pronunciation	Context
(1) lɛd	... it monitors the *lead* levels in drinking ...
" "	... conference on *lead* poisoning in ...
" "	... strontium and *lead* isotope zonation ...
(2) li:d	... maintained their *lead* Thursday over ...
" "	... to Boston and *lead* singer for Purple ...
" "	... Bush a 17-point *lead* in Texas , only 3 ...

Test Data:

Pronunciation	Context
???	... median blood *lead* concentration was ..
???	... his double-digit *lead* nationwide . The ...

p(class | token in context)

Spelling Correction

Problem:

... and he fired presidential **aid/aide** Dick Morris after ...

⇒ *aid* or

⇒ *aide*

Training Data:

Spelling	Context
(1) aid	... and cut the foreign *aid/aide* budget in fiscal 1996 ...
" "	... they offered federal *aid/aide* for flood-ravaged states ...
(2) aide	... fired presidential *aid/aide* Dick Morris after ...
" "	... and said the chief *aid/aide* to Sen. Baker, Mr. John ...

Test Data:

Spelling	Context
???	... said the longtime *aid/aide* to the Mayor of St. ...
???	... will squander the *aid/aide* it receives from the ...

Slide courtesy Jason Eisner, with mild edits

slide courtesy of D. Yarowsky

What features? Example: “word to [the] left [of correction]”

Word to left	Frequency as **Aid**	Frequency as **Aide**
foreign	718	1
federal	297	0
western	146	0
provide	88	0
covert	26	0
oppose	13	0
future	9	0
similar	6	0
presidential	0	63
chief	0	40
longtime	0	26
aids-infected	0	2
sleepy	0	1
disaffected	0	1
indispensable	2	1
practical	2	0
squander	1	0

Spelling correction using an n-gram language model ($n \geq 2$) would use words to left and right to help predict the true word.

Similarly, an HMM would predict a word's class using classes to left and right.

But we'd like to throw in all kinds of other features, too …

An assortment of possible cues ...

	Position	Collocation	lɛd	li:d
N-grams	+1 L	lead *level/N*	219	0
	-1 W	*narrow* lead	0	70
(word,	+1 W	lead *in*	207	898
lemma,	-1W,+1W	*of* lead *in*	162	0
part-of-speech)	-1W,+1W	*the* lead *in*	0	301
	+1P,+2P	lead , <*NOUN*>	234	7
Wide-context	±k W	*zinc* (in ±*k* words)	235	0
collocations	±k W	*copper* (in ±*k* words)	130	0
Verb-object	-V L	*follow/V* + lead	0	527
relationships	-V L	*take/V* + lead	1	665

generates a whole bunch of potential cues – use data to find out which ones work best

Word to left	Frequency as **Aid**	Frequency as **Aide**
foreign	718	1
federal	297	0
western	146	0
provide	88	0

An assortment of possible cues ...

	Position	Collocation	lɛd	li:d
N-grams	+1 L	lead *level/N*	219	0
	-1 W	*narrow* lead	0	70
(word,	+1 W	lead *in*	207	898
lemma,	-1W,+1W	*of* lead *in*	162	0
part-of-speech)	-1W,+1W	*the* lead *in*	0	301
	+1P,+2P	lead , <*NOUN*>	234	7
Wide-context	$\pm$k W	*zinc* (in $\pm k$ words)	235	0
collocations	$\pm$k W	*copper* (in $\pm k$ words)	130	0
Verb-object	-V L	*follow/V* + lead	0	527
relationships	-V L	*take/V* + lead	1	665

This feature is relatively weak, but weak features are still useful, especially since very few features will fire in a given context.

merged ranking of all cues of all these types

11.40	*follow/V* + lead	$\Rightarrow$ li:d
11.20	*zinc* (in $\pm k$ words)	$\Rightarrow$ lɛd
11.10	lead *level/N*	$\Rightarrow$ lɛd
10.66	*of* lead *in*	$\Rightarrow$ lɛd
10.59	*the* lead *in*	$\Rightarrow$ li:d
10.51	lead *role*	$\Rightarrow$ li:d

Final decision list for ***lead*** (abbreviated)

List of all features, ranked by their weight.

(These weights are for a simple "decision list" model where the single highest-weighted feature that fires gets to make the decision all by itself.

However, a log-linear model, which adds up the weights of all features that fire, would be roughly similar.)

LogL	Evidence	Pronunciation
11.40	*follow/V* + lead	⇒ li:d
11.20	*zinc* (in $\pm k$ words)	⇒ lɛd
11.10	lead *level/N*	⇒ lɛd
10.66	*of* lead *in*	⇒ lɛd
10.59	*the* lead *in*	⇒ li:d
10.51	lead *role*	⇒ li:d
10.35	*copper* (in $\pm k$ words)	⇒ lɛd
10.28	lead *time*	⇒ li:d
10.24	lead *levels*	⇒ lɛd
10.16	lead *poisoning*	⇒ lɛd
8.55	*big* lead	⇒ li:d
8.49	*narrow* lead	⇒ li:d
7.76	*take/V* + lead	⇒ li:d
5.99	lead , *NOUN*	⇒ lɛd
$\boxed{1.15}$	lead *in*	⇒ li:d
	∘ ∘ ∘	

Slide courtesy Jason Eisner, with mild edits

Text Annotation Tasks

1. Classify the entire document ("text categorization")
2. Classify individual word tokens
3. Classify word tokens in a sequence
4. Identify phrases ("chunking")
5. Syntactic annotation (parsing)
6. Semantic annotation
7. Text generation

Slide courtesy Jason Eisner, with mild edits

Part of Speech Tagging

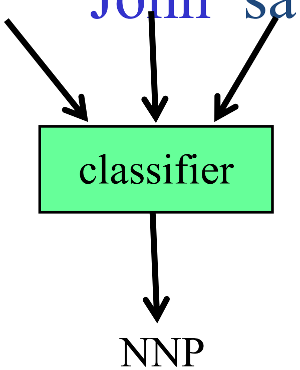
- We could treat tagging as a token classification problem
  - Tag each word independently given features of context
  - And features of the word's spelling (suffixes, capitalization)

Slide courtesy Jason Eisner, with mild edits

Sequence Labeling as Classification

Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.



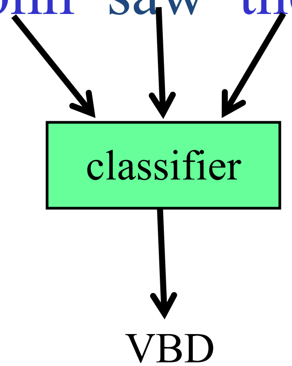
Slide from Ray Mooney

Slide courtesy Jason Eisner, with mild edits

Sequence Labeling as Classification

Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.



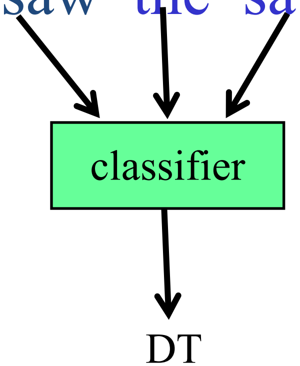
Slide from Ray Mooney

Slide courtesy Jason Eisner, with mild edits

Sequence Labeling as Classification

Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.

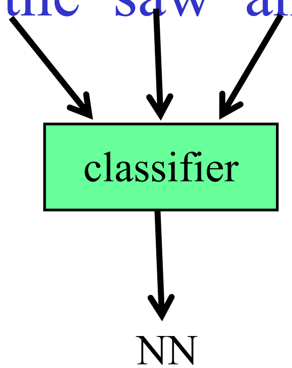


Slide from Ray Mooney

Sequence Labeling as Classification

Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

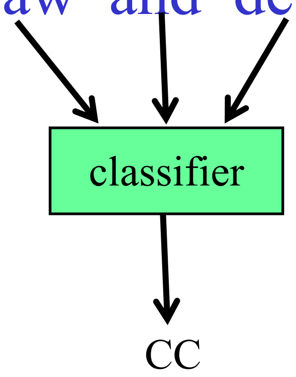
John saw the saw and decided to take it to the table.



Sequence Labeling as Classification

Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.

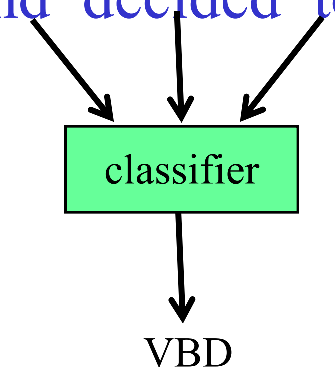


Slide from Ray Mooney

Sequence Labeling as Classification

Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

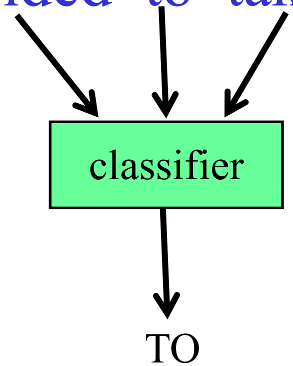
John saw the saw and decided to take it to the table.



Sequence Labeling as Classification

Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.



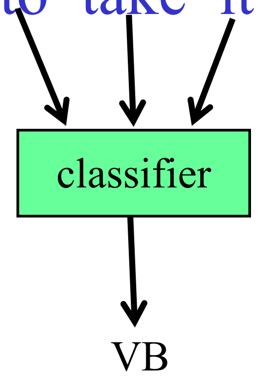
Slide from Ray Mooney

Slide courtesy Jason Eisner, with mild edits

Sequence Labeling as Classification

Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

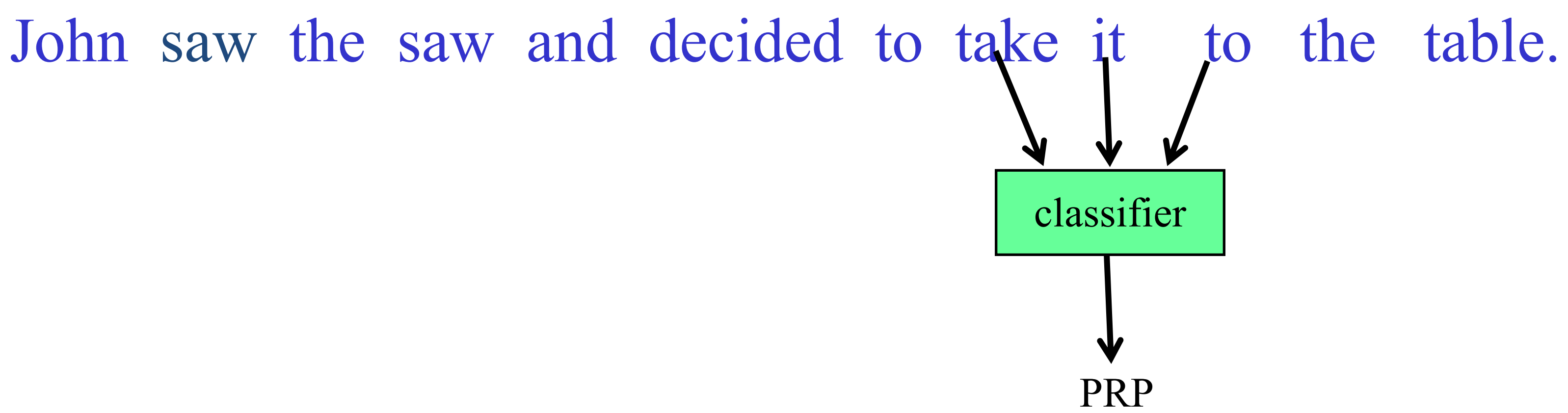
John saw the saw and decided to take it to the table.



Slide from Ray Mooney

Sequence Labeling as Classification

Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

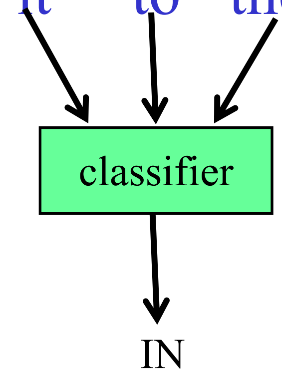


Slide from Ray Mooney

Sequence Labeling as Classification

Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.

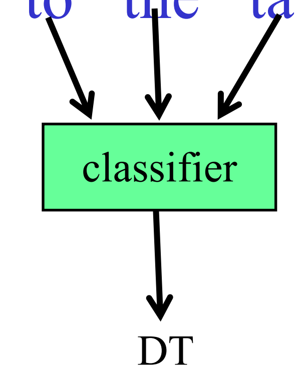


Slide from Ray Mooney

Sequence Labeling as Classification

Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

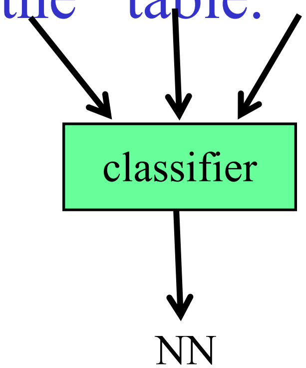
John saw the saw and decided to take it to the table.



Sequence Labeling as Classification

Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

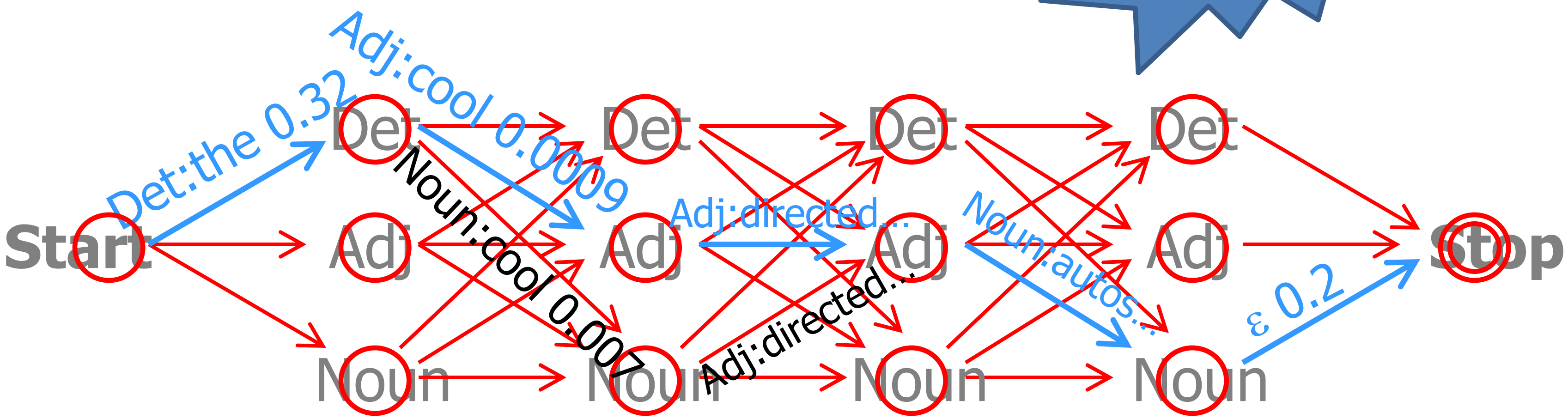
John saw the saw and decided to take it to the table.



Part of Speech Tagging

We'll see HMMs later!

Or we could use an HMM:



probs from tag bigram model

0.4 0.6

Start PN Verb Det Noun Prep Noun Pr

0.001

probs from unigram replacement

Bill directed a cortege of autos thr

Slide courtesy Jason Eisner, with mild edits

Part of Speech Tagging

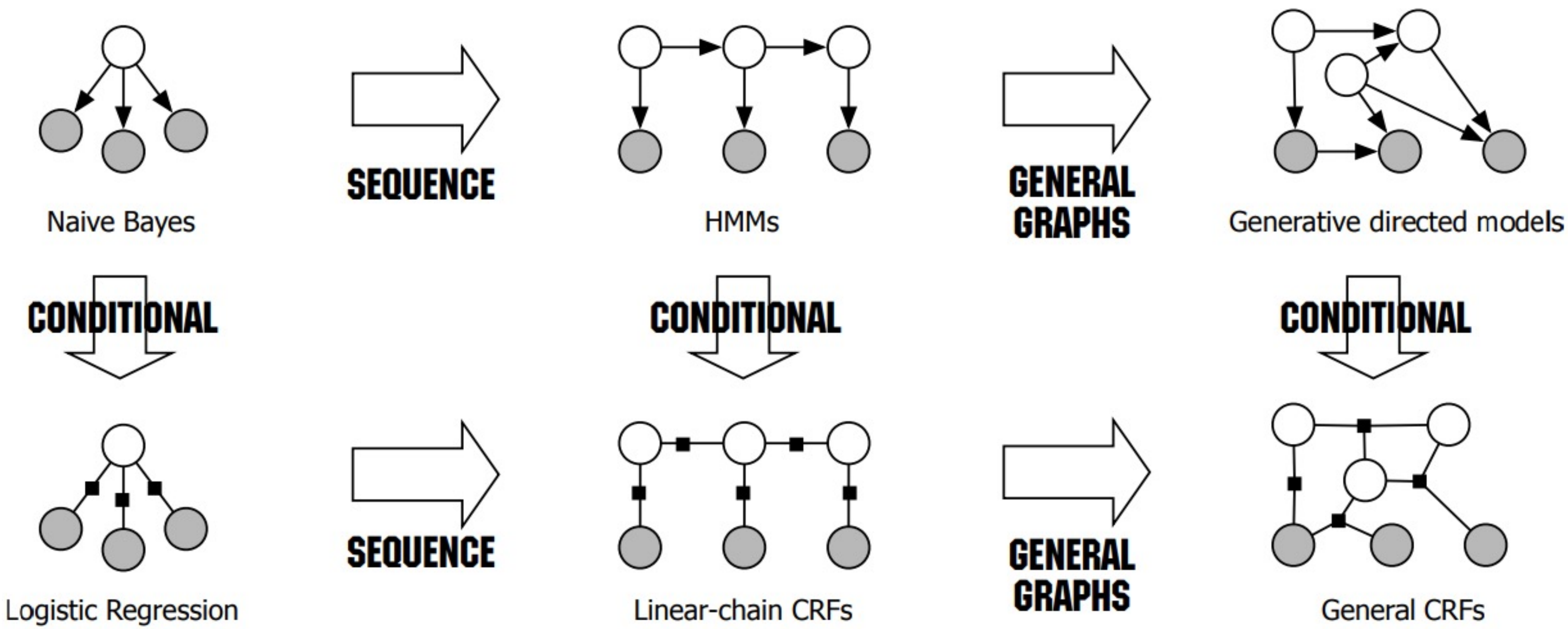
- We could treat tagging as a token classification problem
  - Tag each word independently given features of context
  - And features of the word's spelling (suffixes, capitalization)
- Or we could use an HMM:
  - The point of the HMM is basically that the tag of one word might depend on the tags of adjacent words.
- Combine these two ideas??
  - We'd like rich features (e.g., in a log-linear model), but we'd also like our feature functions to depend on adjacent tags.
  - So, the problem is to predict **all** tags together.

Slide courtesy Jason Eisner, with mild edits

Supervised Learning Methods

- Easy to build a "yes" or "no" predictor from supervised training data
  - Plenty of software packages to do the learning & prediction
  - Lots of people in NLP never go beyond this ☺
- Similarly, easy to build a system that chooses from a small finite set
  - Basically the same deal
  - But runtime goes up linearly with the size of the set, unless you're clever (HW3)
- Harder to predict the best string or tree (set is exponentially large or infinite)

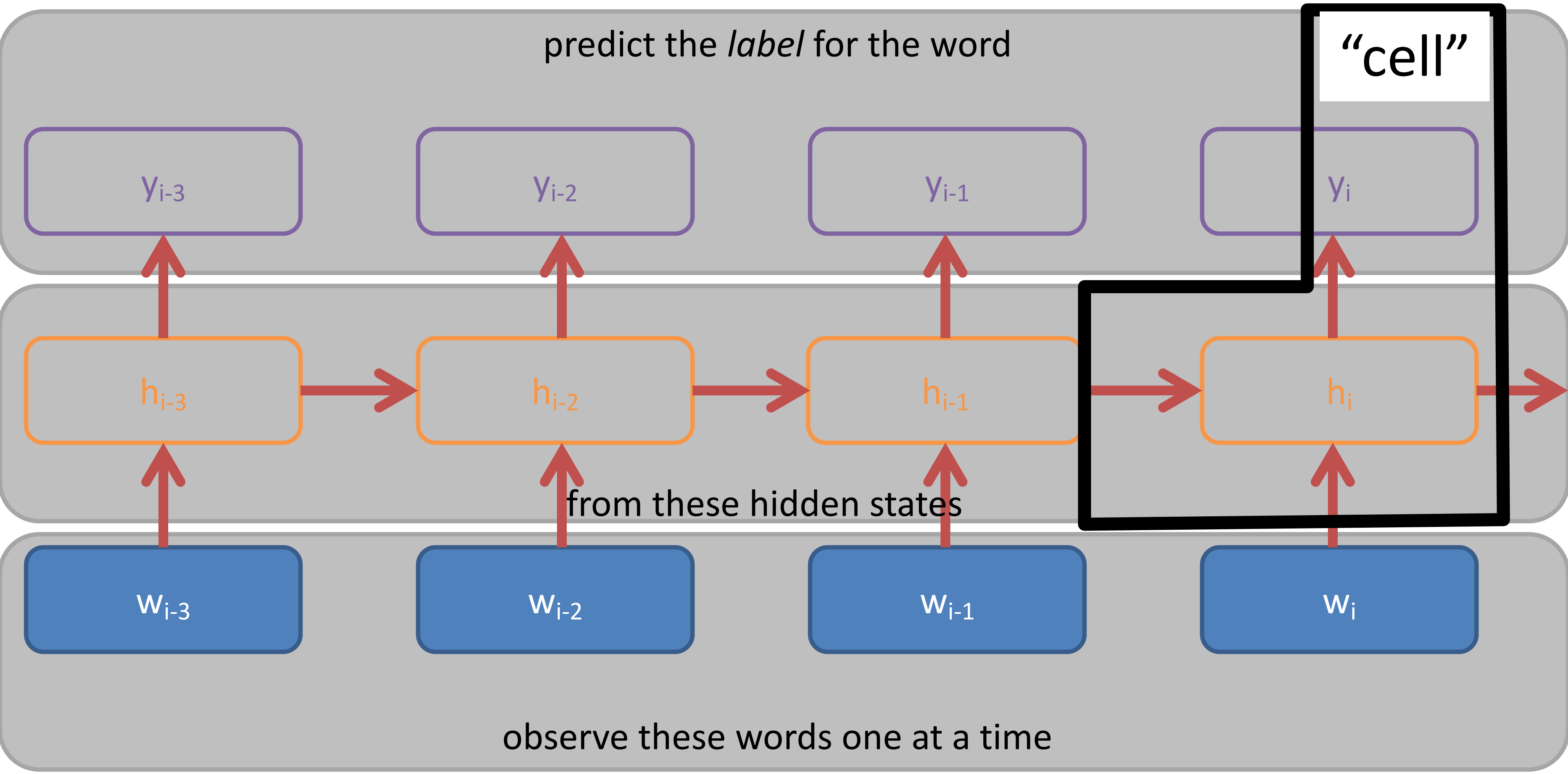
Can We {Do Better?
{Be More Expressive?



See: CMSC 678 or CMSC 691 (Prob. & Graphical ML)

CRF Tutorial, Fig 1.2, Sutton & McCallum (2012)

Can We Use Neural, Recurrent Methods?



Supervised Learning Methods

- Easy to build a "yes" or "no" predictor from supervised training data
  - Plenty of software packages to do the learning & prediction
  - Lots of people in NLP never go beyond this ☺
- Similarly, easy to build a system that chooses from a small finite set
  - Basically the same deal
  - But runtime goes up linearly with the size of the set, unless you're clever (HW3)
- Harder to predict the best string or tree (set is exponentially large or infinite)
  - In the best case, requires dynamic programming; you might have to write your own code
  - But finite-state or CRF toolkits may find the best string for you
  - And you could modify someone else's parser to pick the best tree
  - An algorithm for picking the best can usually be turned into a learning algorithm
  - You may need to rely on **approximate** solutions (e.g., via **beam search**)

Slide courtesy Jason Eisner, with mild edits

Text Annotation Tasks

1. Classify the entire document ("text categorization")
2. Classify individual word tokens
3. Classify word tokens in a sequence
4. Identify phrases ("chunking")
5. Syntactic annotation (parsing)
6. Semantic annotation
7. Text generation

Slide courtesy Jason Eisner, with mild edits

Example: Finding Named Entities

Named entity recognition (NER)

Identify proper names in texts, and classification into a set of predefined categories of interest

Person names

Organizations (companies, government organisations, committees, etc)

Locations (cities, countries, rivers, etc)

Date and time expressions

Measures (percent, money, weight etc), email addresses, Web addresses, street addresses, etc.

Domain-specific: names of drugs, medical conditions, names of ships, bibliographic references etc.

Cunningham and Bontcheva (2003, RANLP Tutorial)

Named Entity Recognition

CHICAGO (AP) — Citing high fuel prices, United Airlines said Friday it has increased fares by $6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit AMR, immediately matched the move, spokesman Tim Wagner said. United, a unit of UAL, said the increase took effect Thursday night and applies to most routes where it competes against discount carriers, such as Chicago to Dallas and Atlanta and Denver to San Francisco, Los Angeles and New York.

NE Types

Type	Tag	Sample Categories
People	PER	Individuals, fictional characters, small groups
Organization	ORG	Companies, agencies, political parties, religious groups, sports teams
Location	LOC	Physical extents, mountains, lakes, seas
Geo-Political Entity	GPE	Countries, states, provinces, counties
Facility	FAC	Bridges, buildings, airports
Vehicles	VEH	Planes, trains, and automobiles

Type	Example
People	*Turing* is often considered to be the father of modern computer science.
Organization	The *IPCC* said it is likely that future tropical cyclones will become more intense.
Location	The *Mt. Sanitas* loop hike begins at the base of *Sunshine Canyon*.
Geo-Political Entity	*Palo Alto* is looking at raising the fees for parking in the University Avenue district.
Facility	Drivers were advised to consider either the *Tappan Zee Bridge* or the *Lincoln Tunnel*.
Vehicles	The updated *Mini Cooper* retains its charm and agility.

Slide from Jim Martin

Example: Information Extraction

As a task: **Filling slots in a database from sub-segments of text.**

October 14, 2002, 4:00 a.m. PT

For years, Microsoft Corporation CEO Bill Gates railed against the economic philosophy of open-source software with Orwellian fervor, denouncing its communal licensing as a "cancer" that stifled technological innovation.

Today, Microsoft claims to "love" the open-source concept, by which software code is made public to encourage improvement and development by outside programmers. Gates himself says Microsoft will gladly disclose its crown jewels--the coveted code behind the Windows operating system--to select customers.

"We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access.“

Richard Stallman, founder of the Free Software Foundation, countered saying…

IE

NAME	TITLE	ORGANIZATION
Bill Gates	CEO	Microsoft
Bill Veghte	VP	Microsoft
Richard Stallman	founder	Free Soft..

Slide from Chris Brew, adapted from slide by William Cohen

Phrase Types to Identify for IE

Closed set

U.S. states

He was born in Alabama…

The big Wyoming sky…

Regular set

U.S. phone numbers

Phone: (413) 545-1323

The CALD main office can be reached at 412-268-1299

Complex pattern

U.S. postal addresses

University of Arkansas
P.O. Box 140
Hope, AR 71802

Headquarters:
1128 Main Street, 4th Floor
Cincinnati, Ohio 45210

Ambiguous patterns, needing context and many sources of evidence

Person names

…was among the six houses sold by Hope Feldman that year.

Pawel Opalinski, Software Engineer at WhizBang Labs.

Slide from Chris Brew, adapted from slide by William Cohen

Slide courtesy Jason Eisner, with mild edits

Identifying phrases

- A key step in IE is to identify relevant phrases
  - Named entities
    - As on previous slides
  - Relationship phrases
    - “said”, “according to”, …
    - “was born in”, “hails from”, …
    - “bought”, “hopes to acquire”, “formed a joint agreement with”, …
  - Simple syntactic chunks (e.g., non-recursive NPs)
    - “Syntactic chunking” sometimes done before (or instead of) parsing
    - Also, “segmentation”: divide Chinese text into words (no spaces)
- So, how do we learn to mark phrases?

Slide courtesy Jason Eisner, with mild edits

Reduce to a tagging problem …

- The IOB encoding (Ramshaw & Marcus 1995):
  - B_X = "beginning" (first word of an X)
  - I_X = "inside" (non-first word of an X)
  - O = "outside" (not in any phrase)
  - Does not allow overlapping or recursive phrases

…United Airlines said Friday it has increased …

B_ORG I_ORG O O O O O

… the move , spokesman Tim Wagner said …

O O O O B_PER I_PER O

What if this were tagged as B_ORG instead?

Example applications for IE

- Classified ads
- Restaurant reviews
- Bibliographic citations
- Appointment emails
- Legal opinions
- Papers describing clinical medical studies
- …

Text Annotation Tasks

1. Classify the entire document ("text categorization")
2. Classify individual word tokens
3. Classify word tokens in a sequence
4. Identify phrases ("chunking")
5. Syntactic annotation (parsing)
6. Semantic annotation
7. Text generation

Slide courtesy Jason Eisner, with mild edits

Garden Path Sentences

The old man the boat .

Garden Path Sentences

The old man the boat .

Garden Path Sentences

The complex houses married and single soldiers and their families.

Garden Path Sentences

The complex houses married and single soldiers and their families.

Garden Path Sentences

The rat the cat the dog chased killed ate the malt.

Garden Path Sentences

The rat *that* the cat the dog chased killed ate the malt.

Garden Path Sentences

The rat *that* the cat *that* the dog chased killed ate the malt.

Garden Path Sentences

The rat *that* the cat *that* the dog chased killed ate the malt.

Garden Path Sentences

The rat *that* the cat *that* the dog chased killed ate the malt.

Garden Path Sentences

The rat *that* the cat *that* the dog chased killed ate the malt.

Garden Path Sentences

[The rat [the cat [the dog chased] killed] ate the malt].

Language can have recursive patterns

Syntactic parsing can help identify those

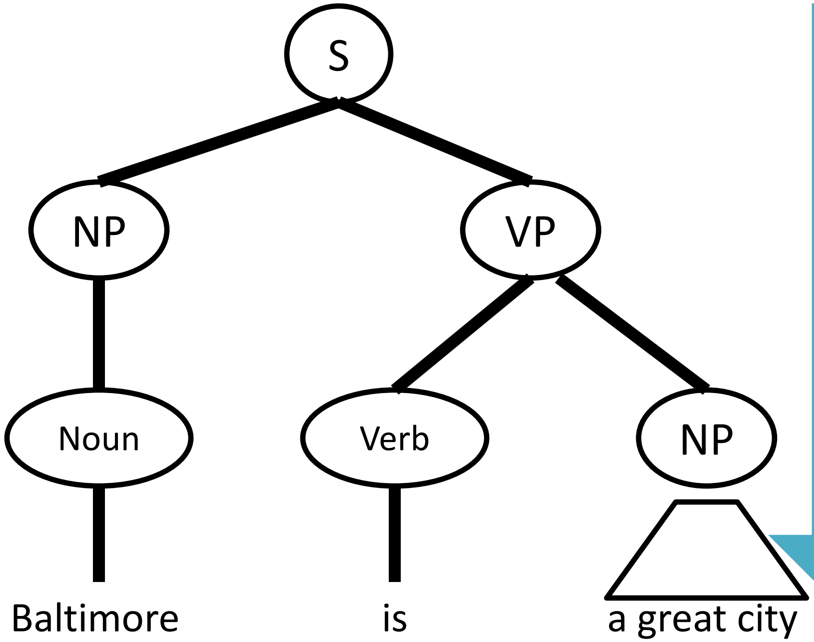
Context Free Grammar

S → NP VP	PP → P NP
NP → Det Noun	AdjP → Adj Noun
NP → Noun	VP → V NP
NP → Det AdjP	Noun → Baltimore
NP → NP PP	…

Set of rewrite rules, comprised of terminals and non-terminals

Generate from a Context Free Grammar

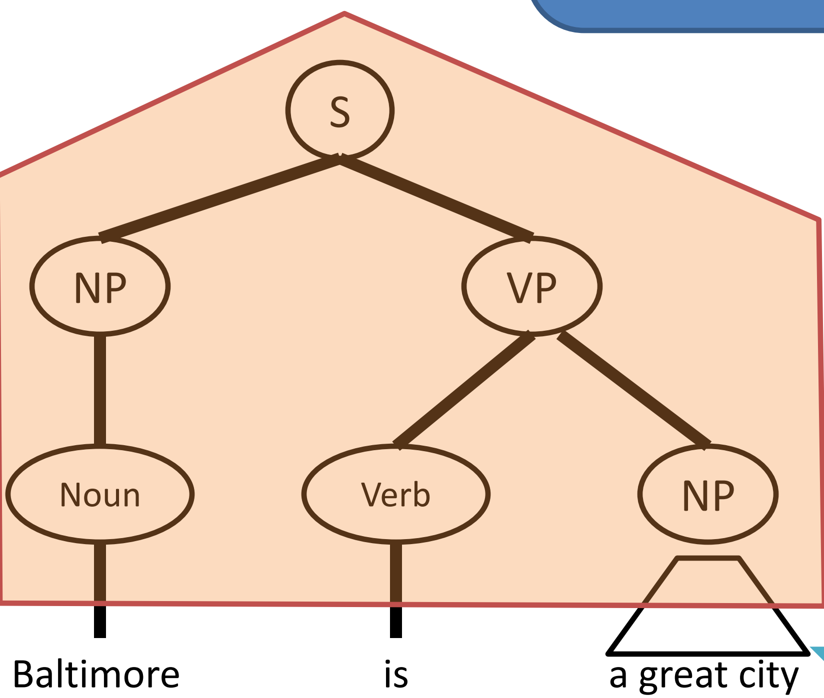
S → NP VP
NP → Det Noun
NP → Noun
NP → Det AdjP
NP → NP PP
PP → P NP
AdjP → Adj Noun
VP → V NP
Noun → Baltimore
…



Baltimore is a great city

Assign Structure (Parse) with a Context Free Grammar

S → NP VP
NP → Det Noun
NP → Noun
NP → Det AdjP
NP → NP PP
PP → P NP
AdjP → Adj Noun
VP → V NP
Noun → Baltimore
…



Baltimore is a great city

$[_{S}\ [_{NP}\ [_{Noun}$ Baltimore]] $[_{VP}\ [_{Verb}$ is] $[_{NP}$ a great city]]]

bracket notation

(S (NP (Noun Baltimore))
(VP (V is)
(NP a great city)))

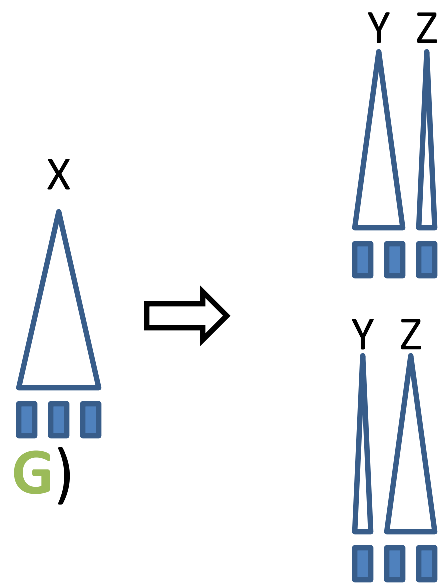
S-expression

You may get to use a **dynamic program**

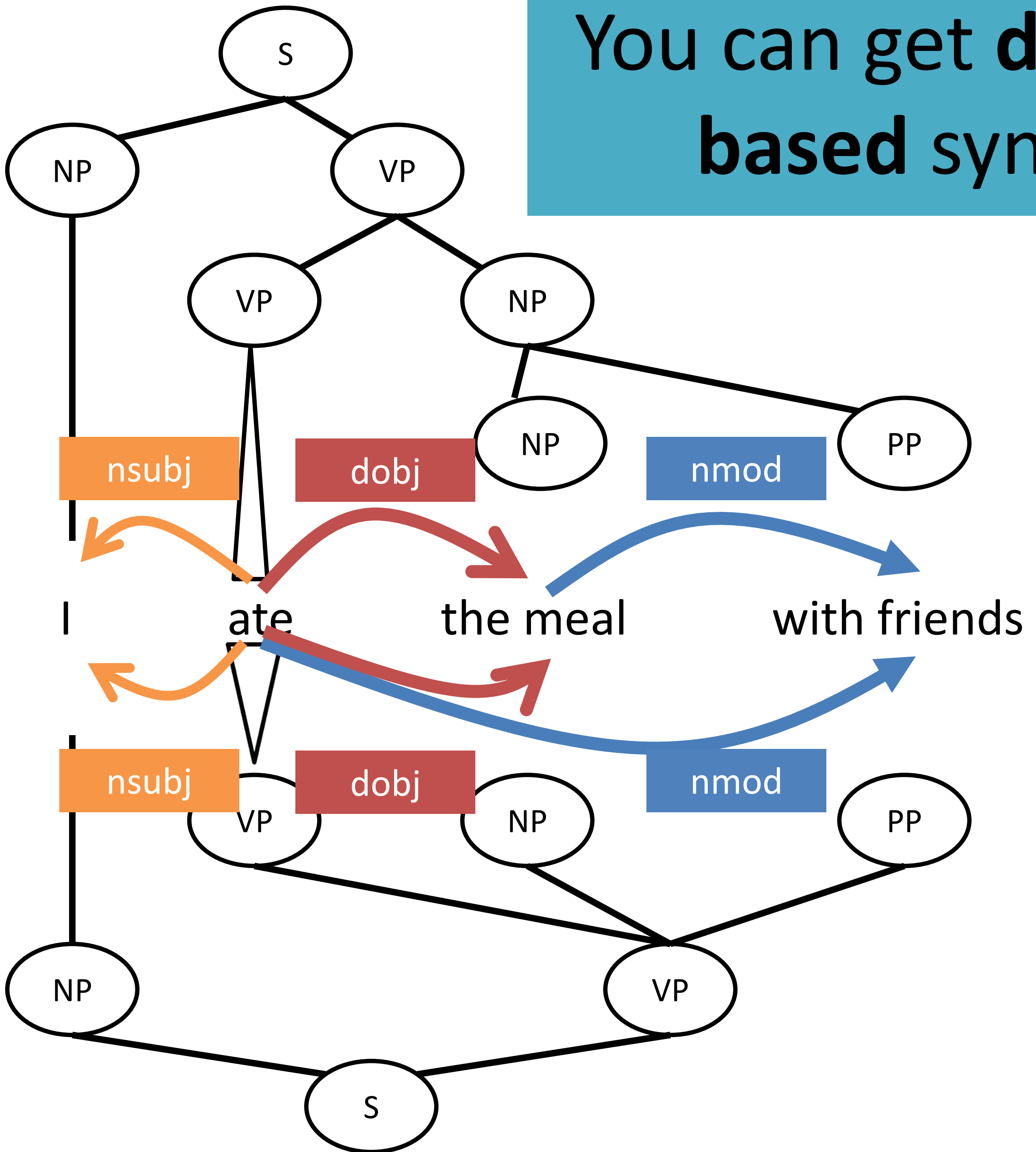
```
T =  Cell[N][N+1]

for(j = 1; j ≤ N; ++j) {
   T[j-1][j].add(X for non-terminal X in G if X → word_j)
}

for(width = 2; width ≤ N; ++width) {
   for(start = 0; start < N - width; ++start) {
      end = start + width
      for(mid = start+1; mid < end; ++mid) {
         for(non-terminal Y : T[start][mid]) {
            for(non-terminal Z : T[mid][end]) {
               T[start][end].add(X for rule X → Y Z : G)
            }
         }
      }
   }
}
```

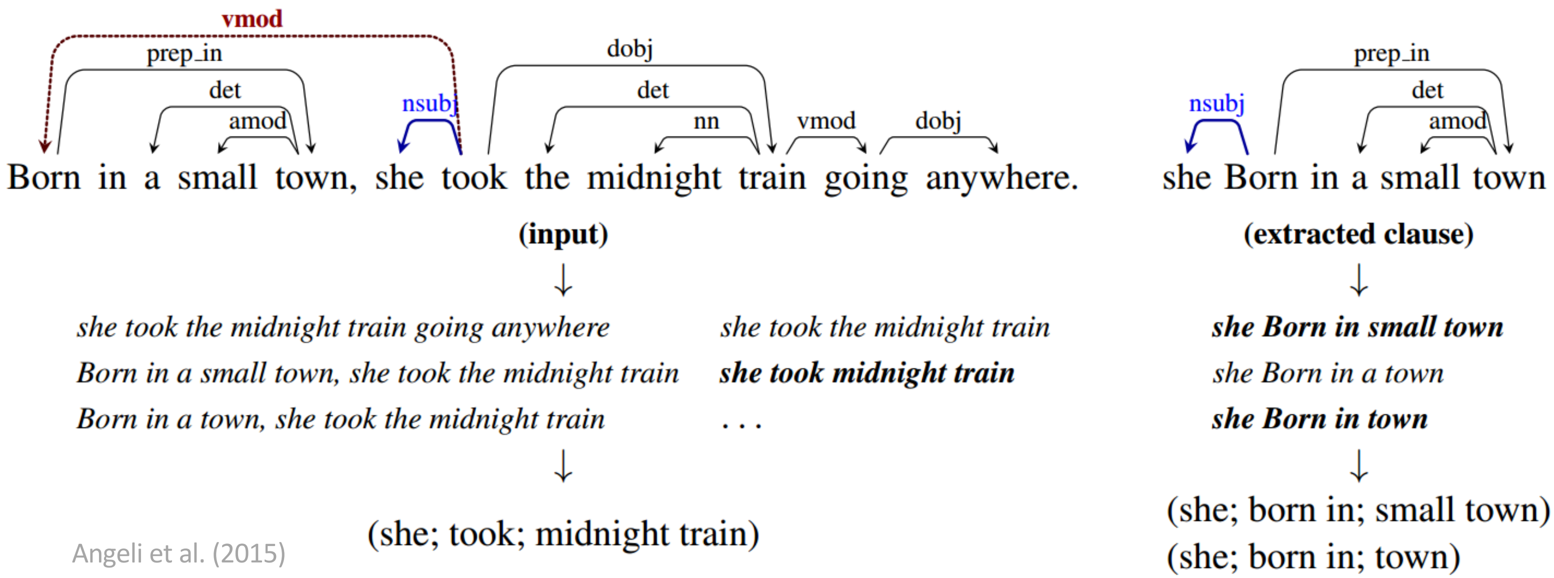


You can get **dependency-based** syntactic forms



And Go From Syntax to Shallow Semantics

"Open Information Extraction"



Angeli et al. (2015)

http://corenlp.run/ (constituency & dependency)

https://github.com/hltcoe/predpatt

a sampling of efforts

http://openie.allenai.org/

http://www.cs.rochester.edu/research/knext/browse/ (constituency trees)

http://rtw.ml.cmu.edu/rtw/

Supervised Learning Methods

- Easy to build a "yes" or "no" predictor from supervised training data
  - Plenty of software packages to do the learning & prediction
  - Lots of people in NLP never go beyond this ☺
- Similarly, easy to build a system that chooses from a small finite set
  - Basically the same deal
  - But runtime goes up linearly with the size of the set, unless you're clever (HW3)
- Harder to predict the best string or tree (set is exponentially large or infinite)
  - In the best case, requires dynamic programming; you might have to write your own code
  - But finite-state or CRF toolkits may find the best string for you
  - And you could modify someone else's parser to pick the best tree
  - An algorithm for picking the best can usually be turned into a learning algorithm
  - You may need to rely on **approximate** solutions (e.g., via **beam search**)
- Hardest if your features look at "non-local" properties of the string or tree
  - Now dynamic programming won't work (or will be something awful like $O(n^9)$)
  - You need some kind of approximate search
  - Can be harder to turn approximate search into a learning algorithm
  - Still, this is a standard preoccupation of machine learning ("structured prediction," "graphical models")

Slide courtesy Jason Eisner, with mild edits

Text Annotation Tasks

1. Classify the entire document ("text categorization")
2. Classify individual word tokens
3. Classify word tokens in a sequence
4. Identify phrases ("chunking")
5. Syntactic annotation (parsing)
6. Semantic annotation
7. Text generation

Slide courtesy Jason Eisner, with mild edits

Semantic Role Labeling (SRL)

- For each <u>predicate</u> (e.g., verb)
  1. find its arguments (e.g., NPs)
  2. determine their **semantic roles**

John <u>drove</u> Mary from Austin to Dallas in his Toyota Prius.

The hammer <u>broke</u> the window.

  - agent: Actor of an action
  - patient: Entity affected by the action
  - source: Origin of the affected entity
  - destination: Destination of the affected entity
  - instrument: Tool used in performing action.
  - beneficiary: Entity for whom action is performed

As usual, can solve as classification …

- Consider one verb at a time: "bit"
- Classify the role (if any) of each of the 3 NPs

Color Code:

not-a-role

agent

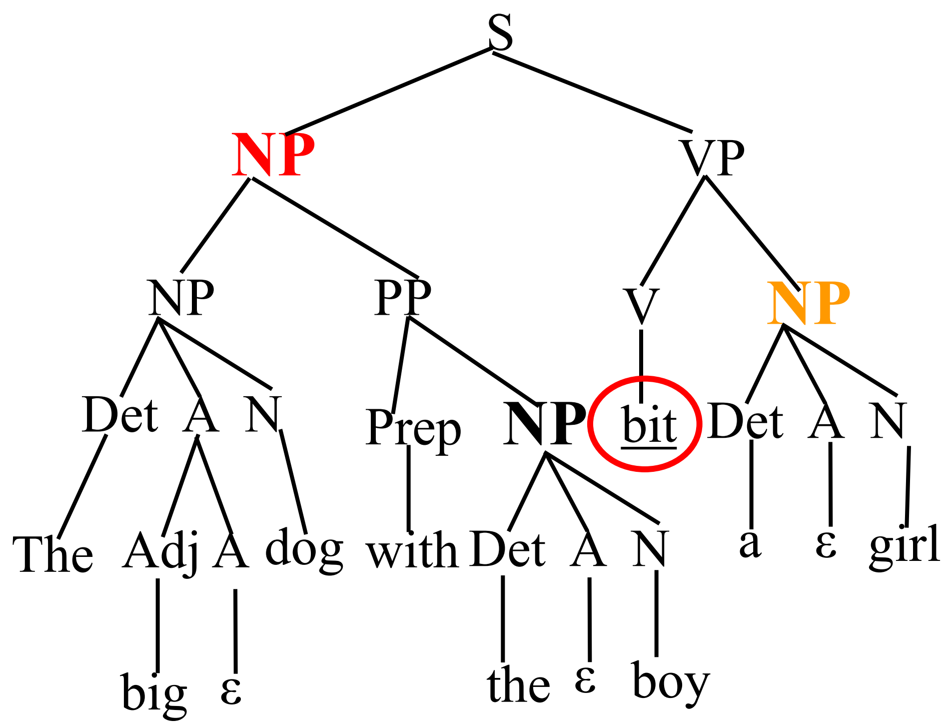
patient

source

destination

instrument

beneficiary

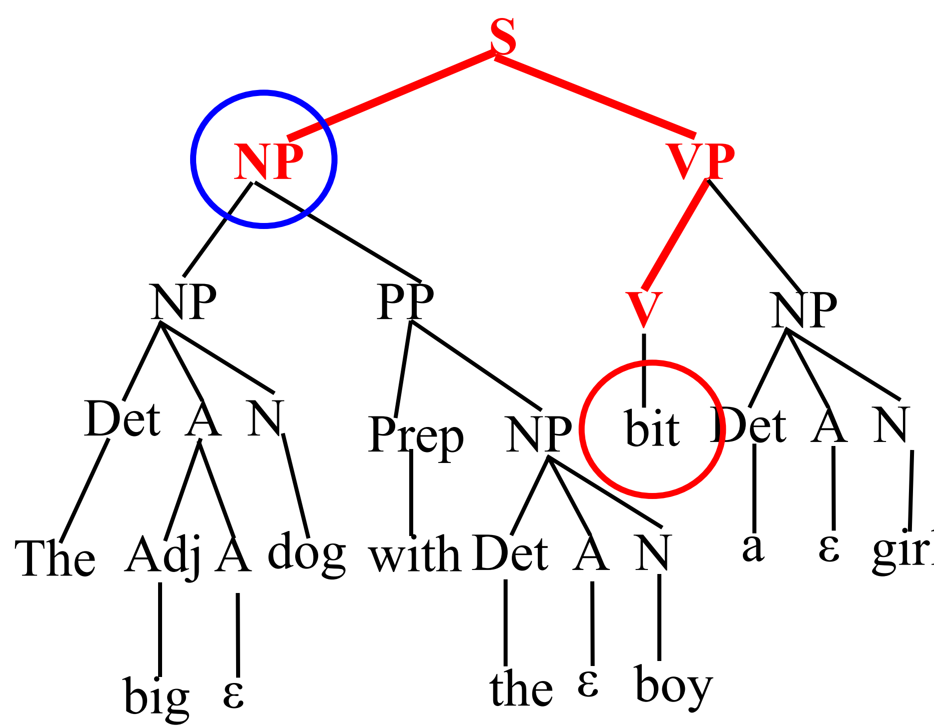


Parse tree paths as classification features

Path feature is

V ↑ VP ↑ S ↓ NP

which tends to be associated with agent role

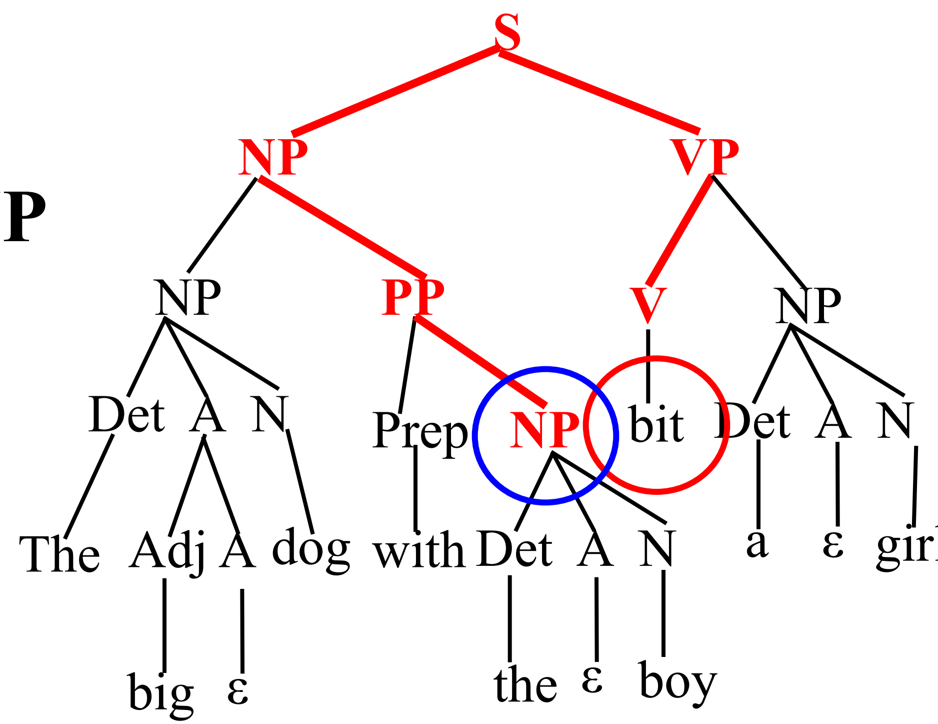


Parse tree paths as classification features

Path feature is

V ↑ VP ↑ S ↓ NP ↓ PP ↓ NP

which tends to be associated with no role



Head words as features

- Some roles prefer to be filled by certain kinds of NPs.
- This can give us useful features for classifying accurately:
  - "John ate the spaghetti with chopsticks." **(instrument)**
    "John ate the spaghetti with meatballs." **(patient)**
    "John ate the spaghetti with Mary."
    - Instruments should be tools
    - Patient of "eat" should be edible
  - "John bought the car for $21K." **(instrument)**
    "John bought the car for Mary." **(beneficiary)**
    - Instrument of "buy" should be Money
    - Beneficiaries should be animate (things with desires)
  - "John drove Mary to school in the van"
    "John drove the van to work with Mary."
    - What do you think?

Uses of Semantic Roles

- Find the answer to a user's question
  - "Who" questions usually want Agents
  - "What" question usually want Patients
  - "How" and "with what" questions usually want Instruments
  - "Where" questions frequently want Sources/Destinations.
  - "For whom" questions usually want Beneficiaries
  - "To whom" questions usually want Destinations
- Generate text
  - Many languages have specific syntactic constructions that must or should be used for specific semantic roles.
- Word sense disambiguation, using selectional restrictions
  - The **bat** ate the **bug**. (what kind of bat? what kind of bug?)
    - Agents (particularly of "eat") should be animate – animal bat, not baseball bat
    - Patients of "eat" should be edible – animal bug, not software bug
  - John **fired** the secretary.

    John **fired** the rifle.

    Patients of fire_1 are different than patients of fire_2

Slide courtesy Jason Eisner, with mild edits

Slide thanks to Ray Mooney (modified)

Other Current Semantic Annotation Tasks
(similar to SRL)

- PropBank – coarse-grained roles of verbs
- NomBank – similar, but for nouns
- FrameNet – fine-grained roles of any word
- TimeBank – temporal expressions

FrameNet Example

REVENGE FRAME

Avenger
Offender (unexpressed in this sentence)
Injury
Injured Party (unexpressed in this sentence)
Punishment

We avenged the insult by setting fire to his village.

a word/phrase that triggers the REVENGE frame

Slide thanks to CJ Fillmore (modified)

Slide courtesy Jason Eisner, with mild edits

FrameNet Example

REVENGE FRAME
triggering words and phrases
(not limited to verbs)

avenge, revenge, retaliate, get back at, pay back, get even, …

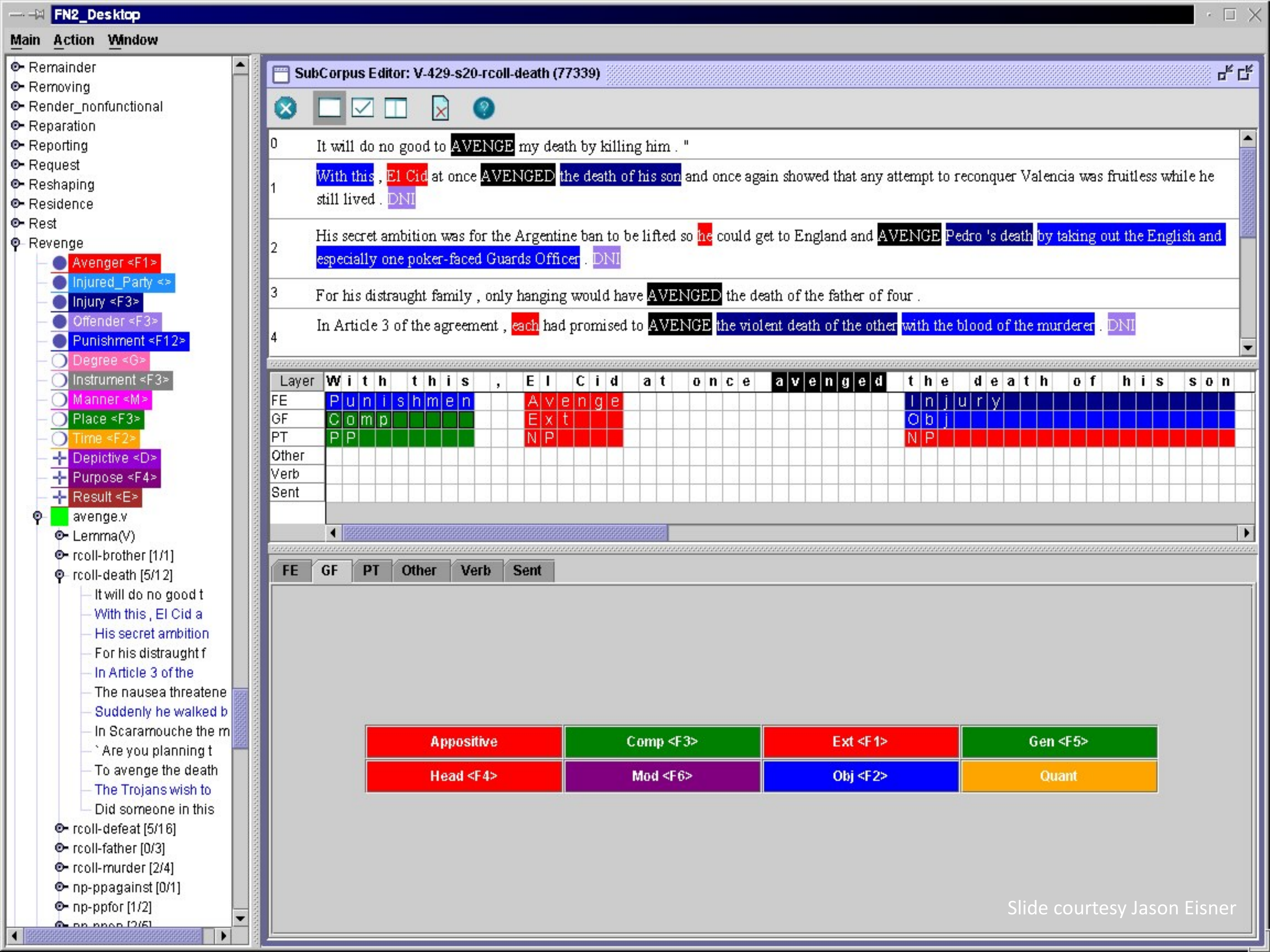
revenge, vengeance, retaliation, retribution, reprisal, …

vengeful, retaliatory, retributive; in revenge, in retaliation, …

take revenge, wreak vengeance, exact retribution, …

Slide thanks to CJ Fillmore (modified)

Slide courtesy Jason Eisner, with mild edits


FN2_Desktop
Main Action Window
Remainder
Removing
Render_nonfunctional
Reparation
Reporting
Request
Reshaping
Residence
Rest
Revenge
Avenger <F1>
Injured_Party <>
Injury <F3>
Offender <F3>
Punishment <F12>
Degree <G>
Instrument <F3>
Manner <M>
Place <F3>
Time <F2>
Depictive <D>
Purpose <F4>
Result <E>
avenge.v
Lemma(V)
rcoll-brother [1/1]
rcoll-death [5/12]
It will do no good t
With this , El Cid a
His secret ambition
For his distraught f
In Article 3 of the
The nausea threatene
Suddenly he walked b
In Scaramouche the m
` Are you planning t
To avenge the death
The Trojans wish to
Did someone in this
rcoll-defeat [5/16]
rcoll-father [0/3]
rcoll-murder [2/4]
np-ppagainst [0/1]
np-ppfor [1/2]
SubCorpus Editor: V-429-s20-rcoll-death (77339)
0 It will do no good to AVENGE my death by killing him . "
1 With this , El Cid at once AVENGED the death of his son and once again showed that any attempt to reconquer Valencia was fruitless while he still lived . DNI
2 His secret ambition was for the Argentine ban to be lifted so he could get to England and AVENGE Pedro 's death by taking out the English and especially one poker-faced Guards Officer . DNI
3 For his distraught family , only hanging would have AVENGED the death of the father of four .
4 In Article 3 of the agreement , each had promised to AVENGE the violent death of the other with the blood of the murderer . DNI
Layer
With this , El Cid at once avenged the death of his son
FE Punishmen Avenge Injury
GF Comp Ext Obj
PT PP NP NP
Other
Verb
Sent
FE GF PT Other Verb Sent
Appositive
Comp <F3>
Ext <F1>
Gen <F5>
Head <F4>
Mod <F6>
Obj <F2>
Quant
Slide courtesy Jason Eisner

Text Annotation Tasks

1. Classify the entire document ("text categorization")
2. Classify individual word tokens
3. Classify word tokens in a sequence
4. Identify phrases ("chunking")
5. Syntactic annotation (parsing)
6. Semantic annotation
7. Text generation

Slide courtesy Jason Eisner, with mild edits

Generating new text

1. Question answering
2. Speech recognition (transcribe as text)
3. Machine translation
4. Text generation from semantics
5. Inflect, analyze, or transliterate words
6. Single- or multi-doc summarization

Deeper Information Extraction

1. Coreference resolution (within a document)
2. Entity linking (across documents)
3. Event extraction and linking
4. Knowledge base population (KBP)
5. Recognizing texual entailment (RTE)

Slide courtesy Jason Eisner, with mild edits

User interfaces

1. Dialogue systems
   - Personal assistance
   - Human-computer collaboration
   - Interactive teaching
2. Language teaching; writing help
3. Question answering
4. Information retrieval

Multimodal interfaces or modeling

1. Sign languages
2. Speech + gestures
3. Images + captions
4. Brain recordings, human reaction times

Slide courtesy Jason Eisner, with mild edits

NLP automates things that humans do well, so that they can be done automatically on more sentences. But this slide is about language analysis that's hard even for humans. Computational linguistics (like comp bio, etc.) can discover underlying patterns in large datasets: things we didn't know!

Discovering Linguistic Structure

1. Decipherment
2. Grammar induction
3. Topic modeling
4. Deep learning of word meanings
5. Language evolution (historical linguistics)
6. Grounded semantics

Slide courtesy Jason Eisner, with mild edits

Today's Learning Goals

- Define featurization and some examples
- Learn about NLP Tasks at a high-level:
  - Document classification
  - Part of speech tagging
  - Syntactic parsing
  - Entity id/coreference