10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

(Multinomial) Logistic Regression + Feature Engineering

Matt Gormley
Lecture 9
Feb. 14, 2018

Reminders

- **Homework 3: KNN, Perceptron, Lin.Reg.**
  - **Out: Wed, Feb 7**
  - **Due: Wed, Feb 14 at 11:59pm**
- **Homework 4: Logistic Regression**
  - **Out: Wed, Feb 14**
  - **Due: Fri, Feb 23 at 11:59pm**

MULTINOMIAL LOGISTIC REGRESSION

Multinomial Logistic Regression

Chalkboard

- Background: Multinomial distribution
- Definition: Multi-class classification
- Geometric intuitions
- Multinomial logistic regression model
- Generative story
- Reduction to binary logistic regression
- Partial derivatives and gradients
- Applying Gradient Descent and SGD
- Implementation w/ sparse features

Debug that Program!

In-Class Exercise: *Think-Pair-Share*

Debug the following program which is (incorrectly) attempting to run SGD for multinomial logistic regression

Buggy Program:

```
while not converged:
  for i in shuffle([1,…,N]):
    for k in [1,…,K]:
       theta[k] = theta[k] - lambda * grad(x[i], y[i], 
theta, k)
```

Assume: `grad(x[i], y[i], theta, k)` returns the gradient of the negative log-likelihood of the training example (x[i],y[i]) with respect to vector `theta[k]`. `lambda` is the learning rate. N = # of examples. K = # of output classes. M = # of features. `theta` is a K by M matrix.

Debug that Program!

In-Class Exercise: *Think-Pair-Share*

Debug the following program which is (incorrectly) attempting to run SGD for multinomial logistic regression

Buggy Program:

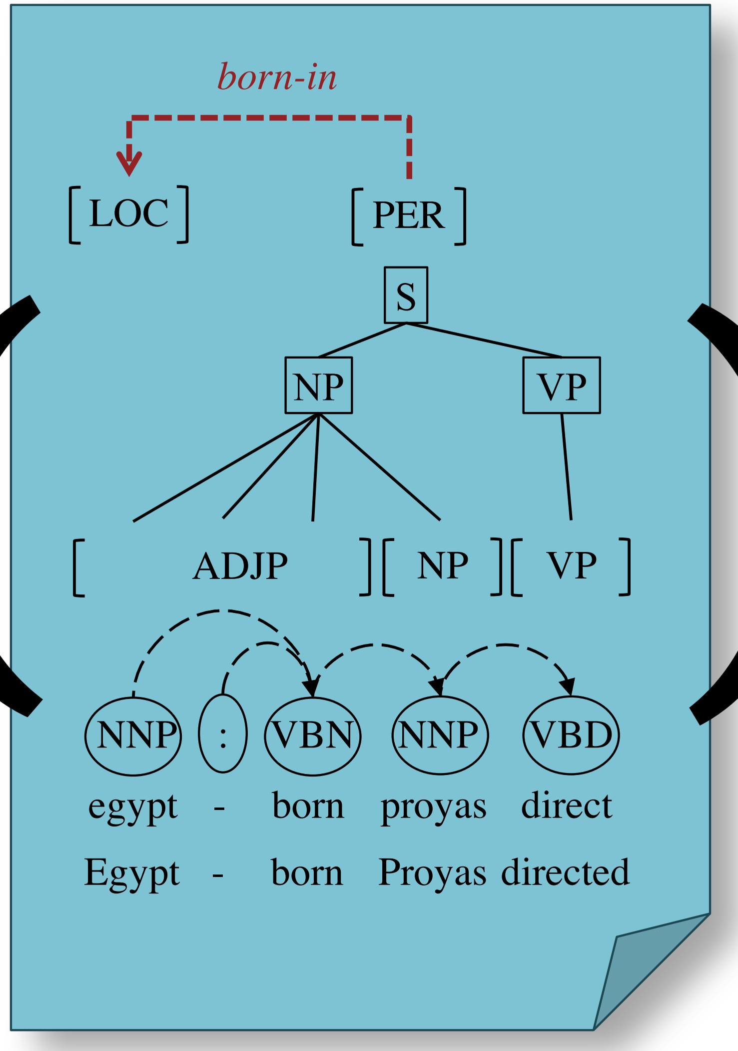
```
while not converged:
  for i in shuffle([1,…,N]):
    for k in [1,…,K]:
      for m in [1,…, M]:
          theta[k,m] = theta[k,m] + lambda * grad(x[i],
y[i], theta, k,m)
```

Assume: `grad(x[i], y[i], theta, k, m)` returns the partial derivative of the negative log-likelihood of the training example (x[i],y[i]) with respect to `theta[k,m]`. `lambda` is the learning rate. N = # of examples. K = # of output classes. M = # of features. `theta` is a K by M matrix.

FEATURE ENGINEERING

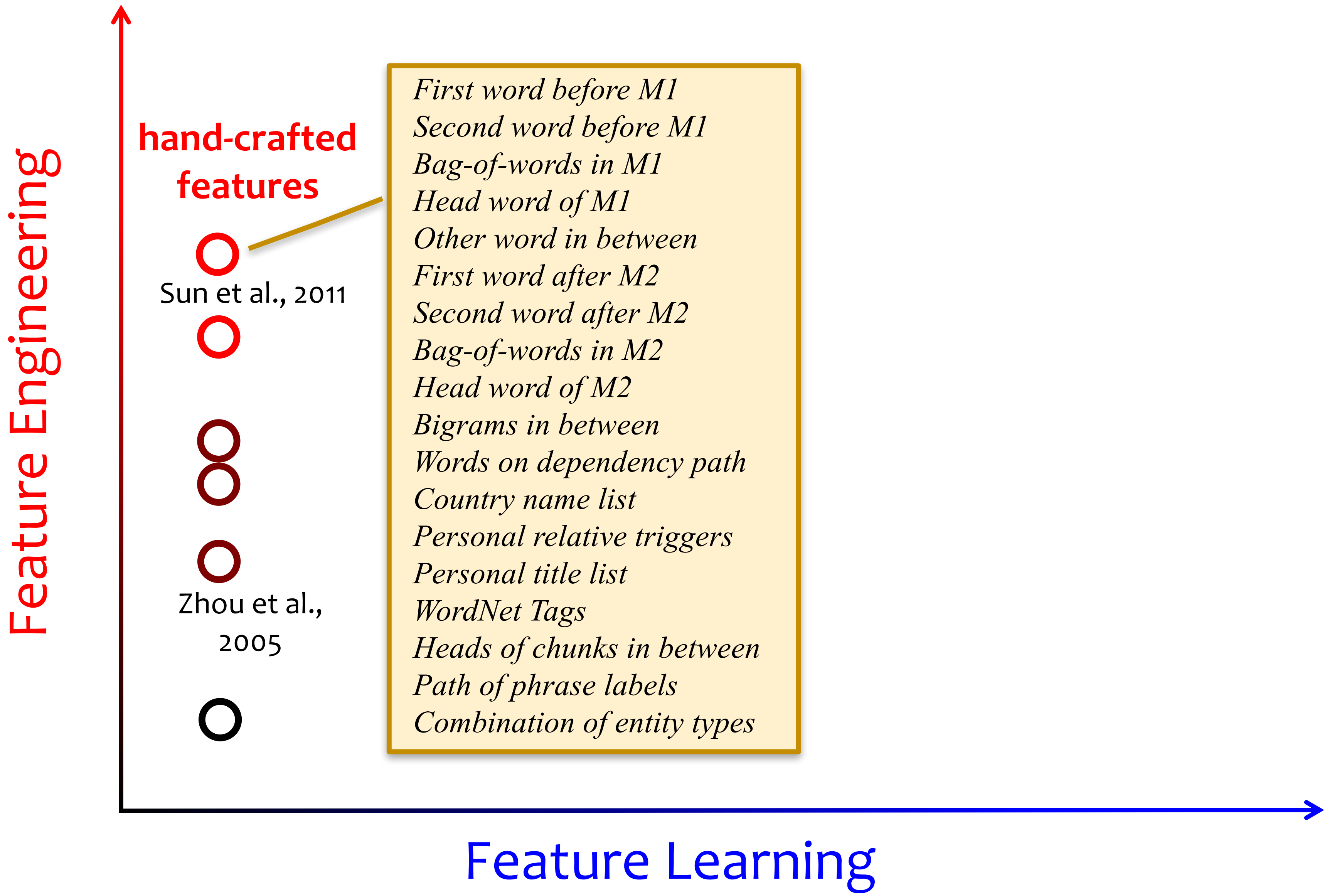
Handcrafted Features

$p(\textcolor{red}{y}|x) \propto \exp(\Theta_{\textcolor{red}{y}} \cdot f\Big($

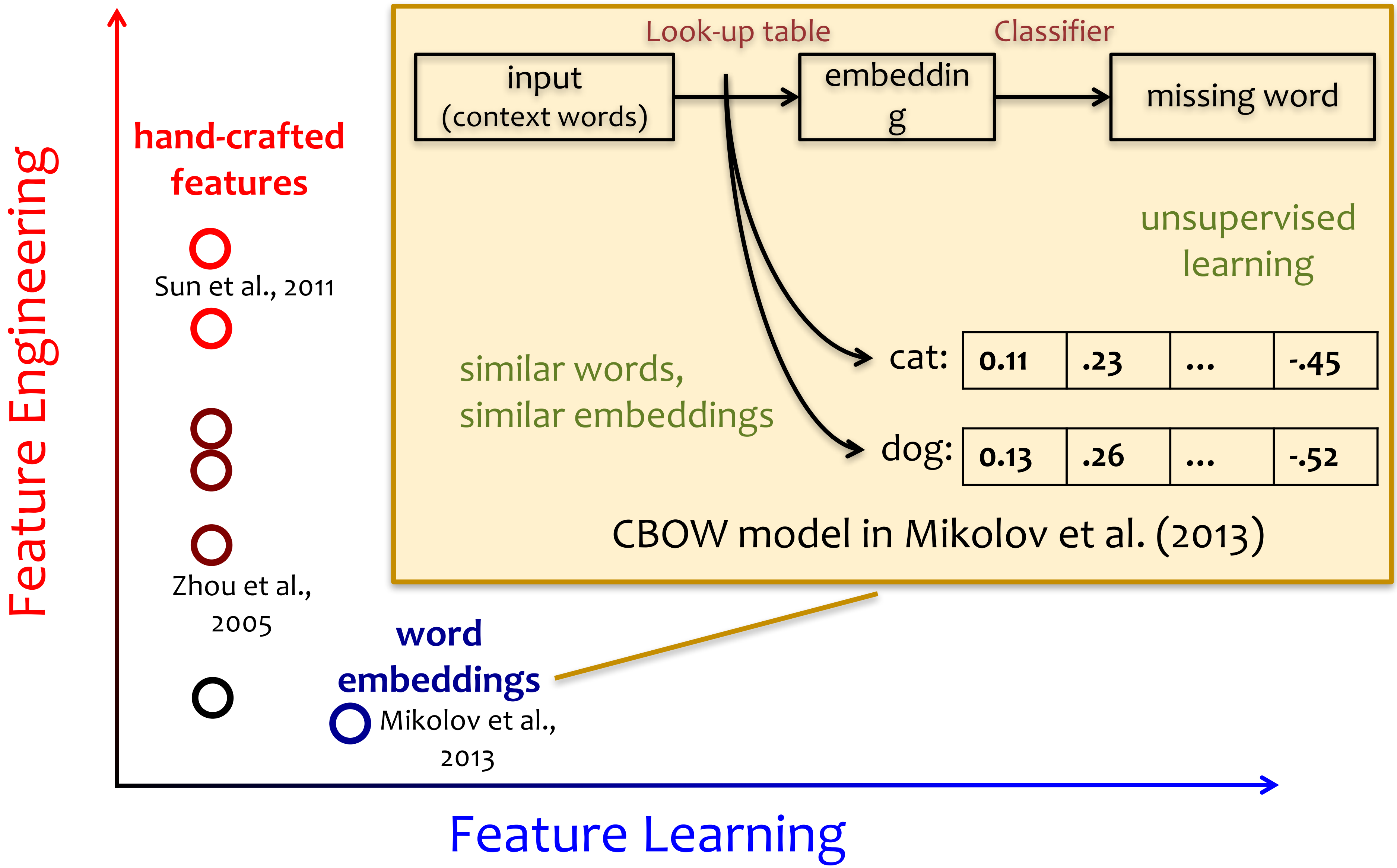


$\Big))$

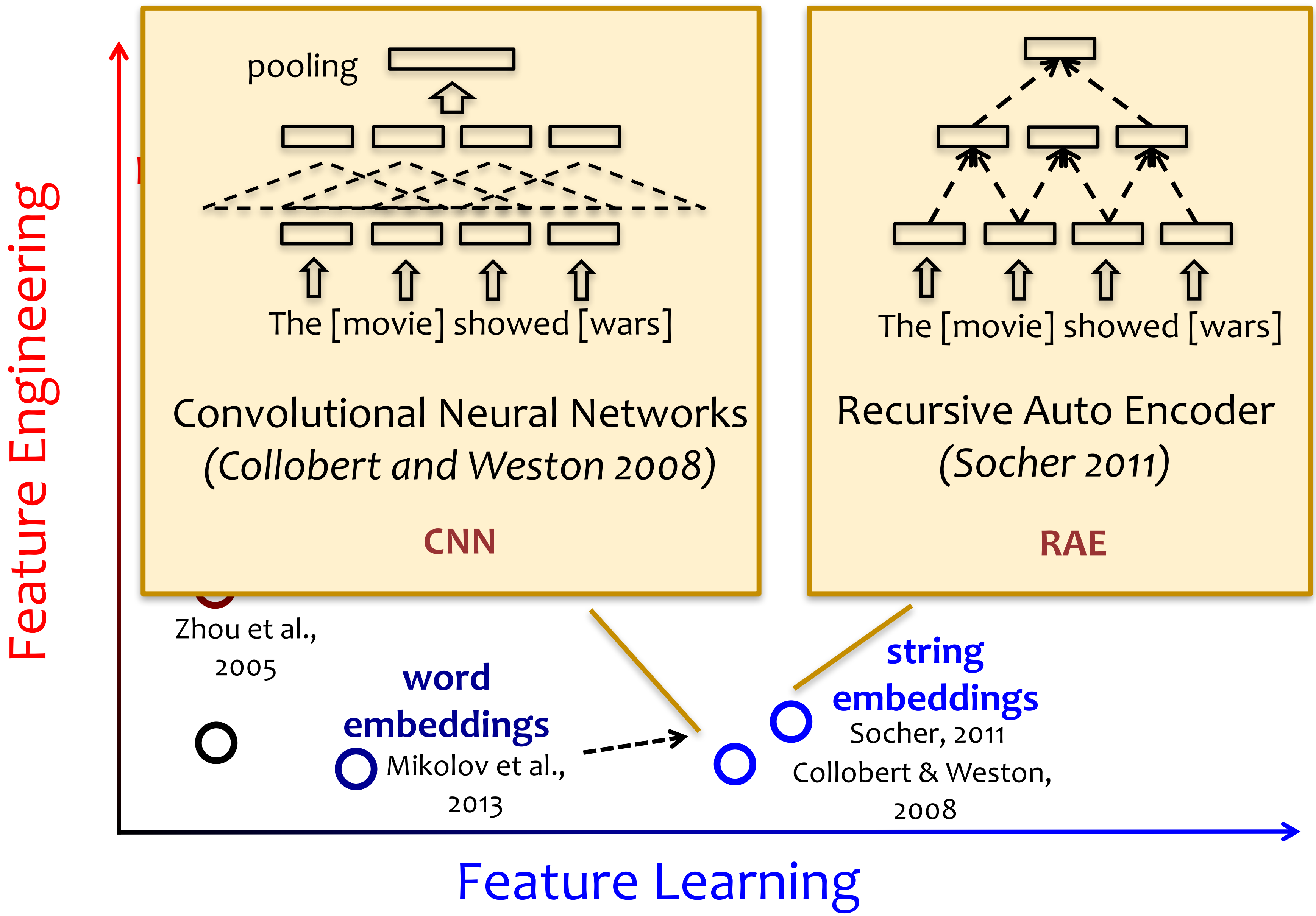
Where do features come from?



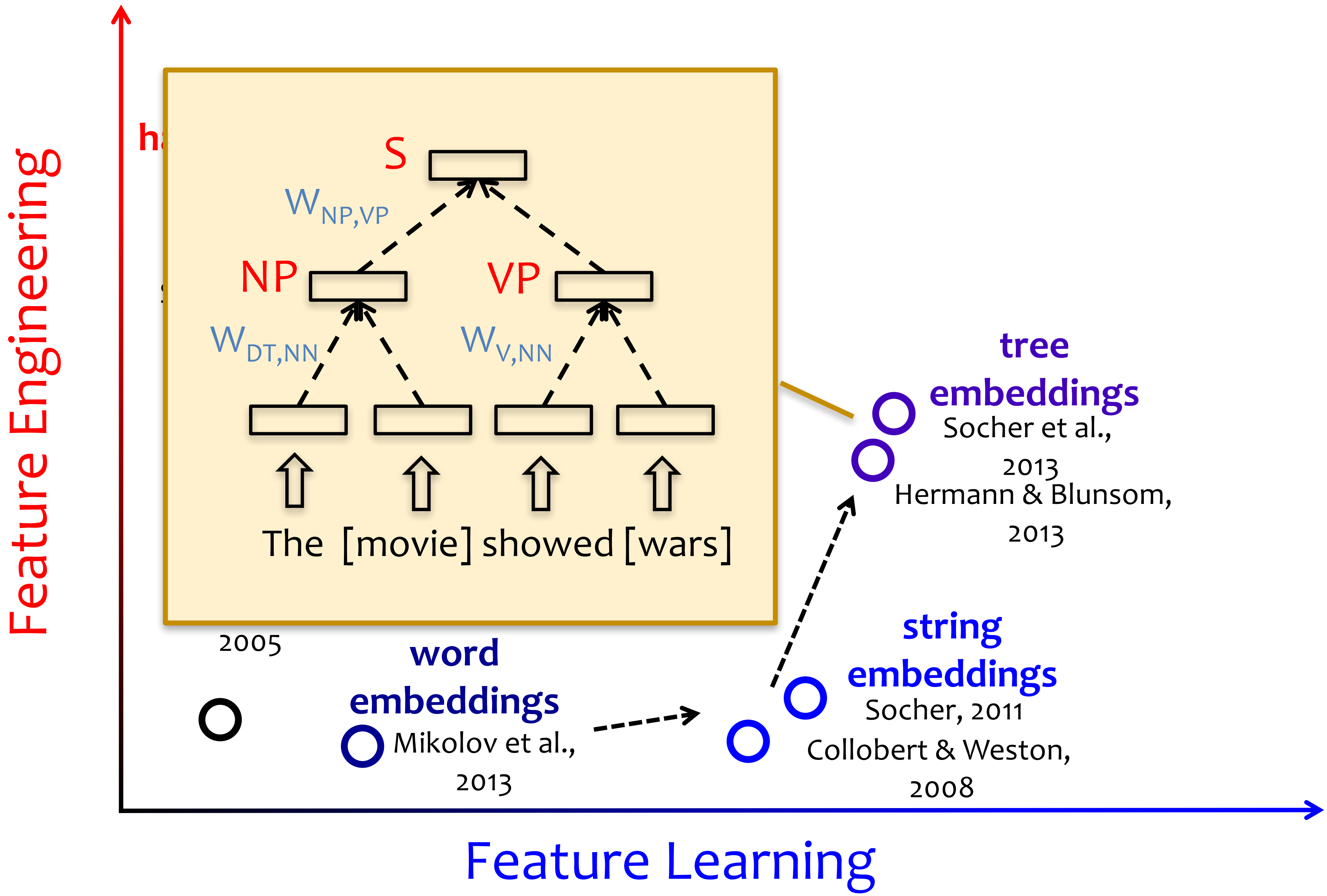
Where do features come from?



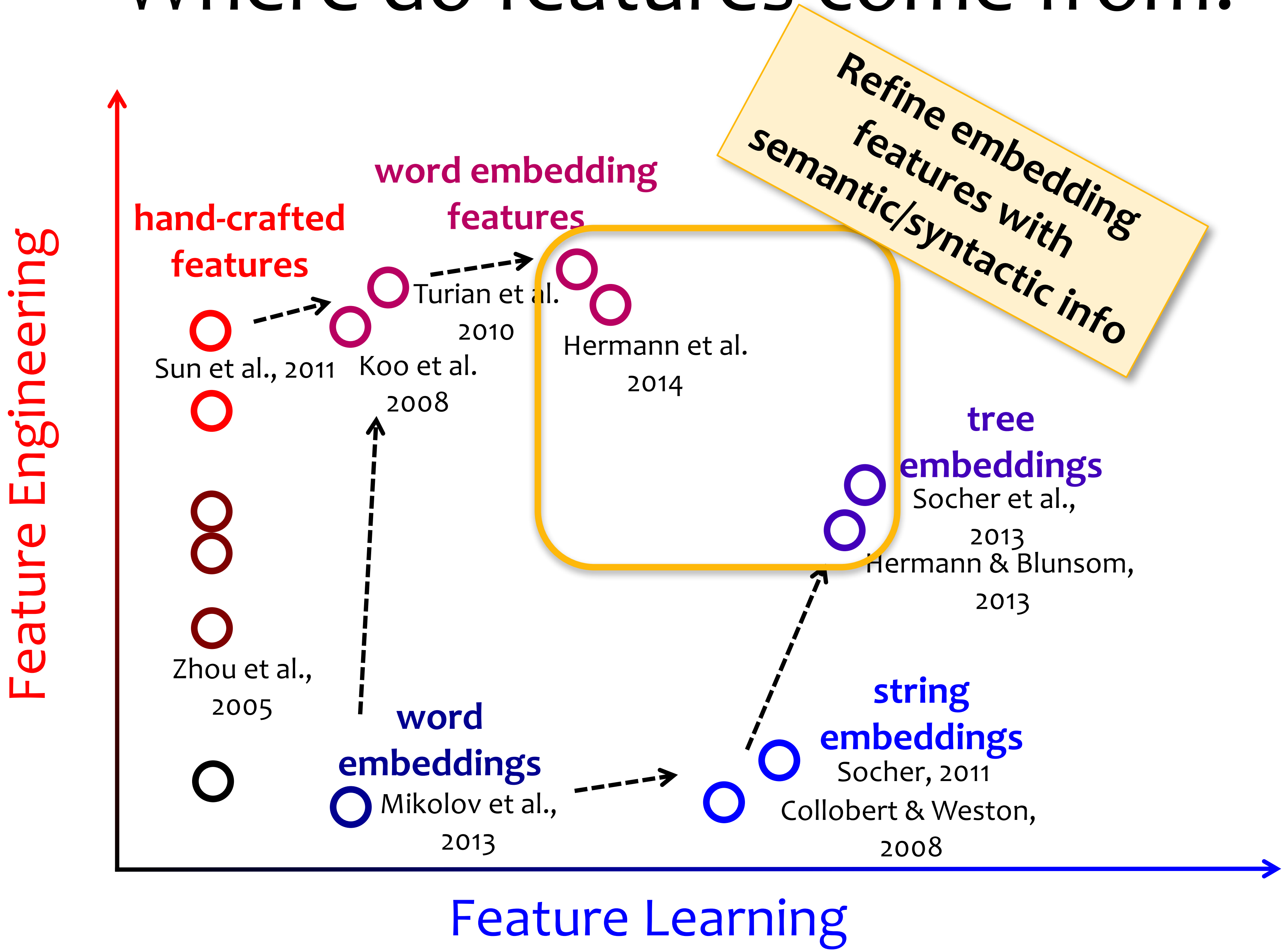
Where do features come from?



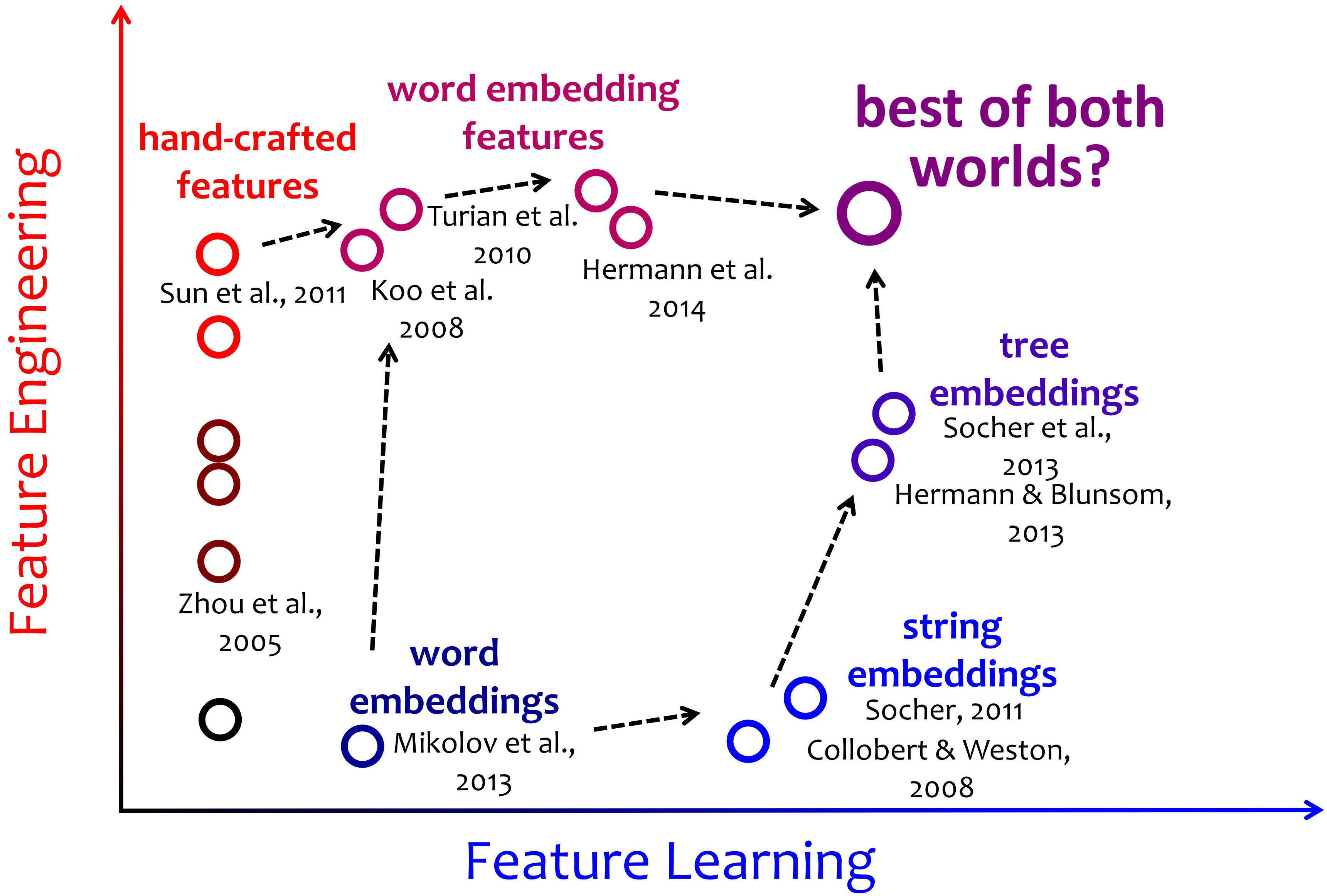
Where do features come from?



Where do features come from?



Where do features come from?



Feature Engineering for NLP

Suppose you build a logistic regression model to predict a part-of-speech (POS) tag for each word in a sentence.

What features should you use?

deter.	noun	noun	verb	verb	noun
The	*movie*	*I*	*watched*	*depicted*	*hope*

Feature Engineering for NLP

Per-word Features:

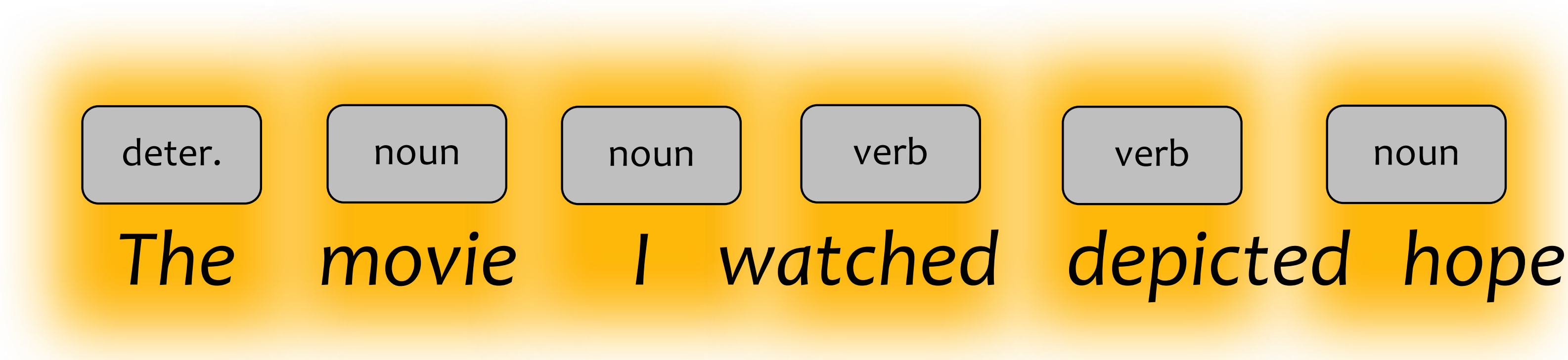
	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
`is-capital(w_i)`	1	0	1	0	0	0
`endswith(w_i,"e")`	1	1	0	0	0	1
`endswith(w_i,"d")`	0	0	0	1	1	0
`endswith(w_i,"ed")`	0	0	0	1	1	0
`w_i == "aardvark"`	0	0	0	0	0	0
`w_i == "hope"`	0	0	0	0	0	1
…	…	…	…	…	…	…

deter.	noun	noun	verb	verb	noun
The	*movie*	*I*	*watched*	*depicted*	*hope*

Feature Engineering for NLP

Context Features:

	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
…	…	…	…	…	…	…
w_i == "watched"	0	0	0	1	0	0
w_{i+1} == "watched"	0	0	1	0	0	0
w_{i-1} == "watched"	0	0	0	0	1	0
w_{i+2} == "watched"	0	1	0	0	0	0
w_{i-2} == "watched"	0	0	0	0	0	1
…	…	…	…	…	…	…



Feature Engineering for NLP

Context Features:

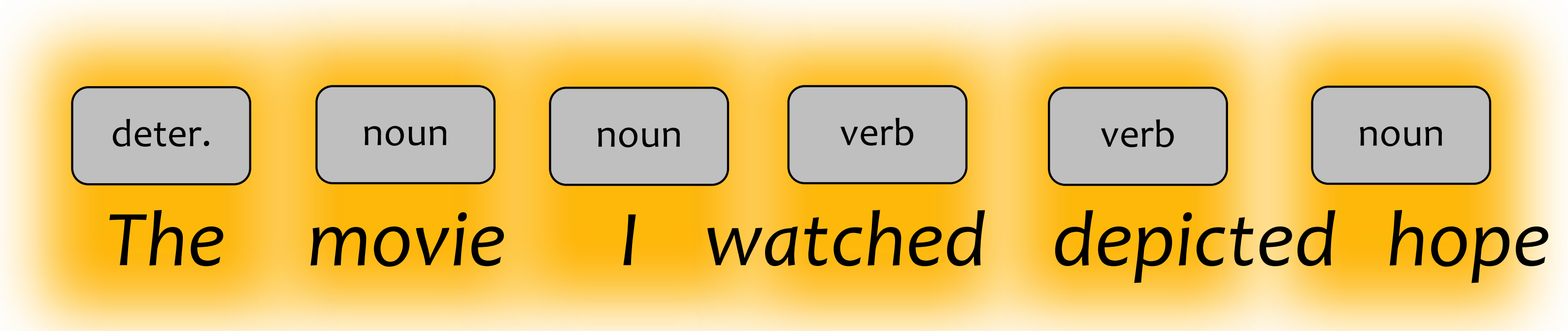
	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
…	…	…	…	…	…	…
`w_i == "I"`	0	0	1	0	0	0
`w_{i+1} == "I"`	0	1	0	0	0	0
`w_{i-1} == "I"`	0	0	0	1	0	0
`w_{i+2} == "I"`	1	0	0	0	0	0
`w_{i-2} == "I"`	0	0	0	0	1	0
…	…	…	…	…	…	…

deter.	noun	noun	verb	verb	noun
The	*movie*	*I*	*watched*	*depicted*	*hope*

Feature Engineering for NLP

Table 3. Tagging accuracies with different feature templates and other changes on the *WSJ* 19-21 development set.

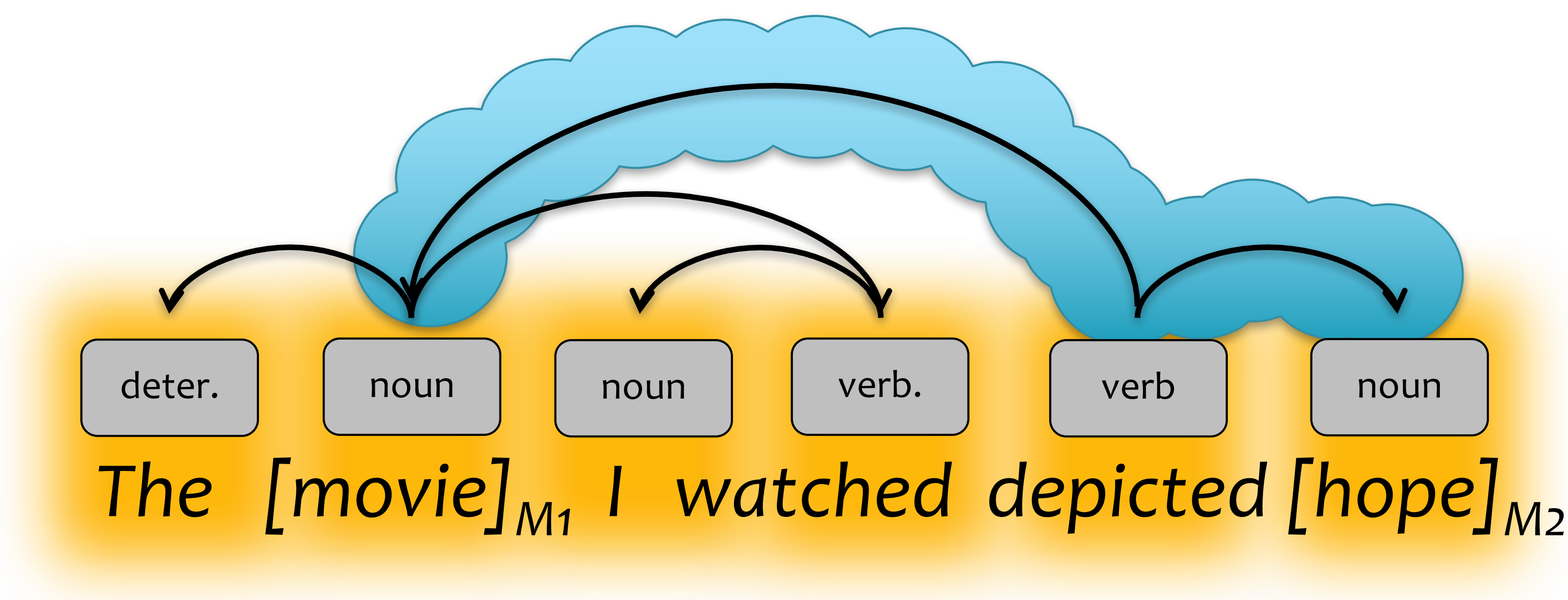
Model	Feature Templates	# Feats	Sent. Acc.	Token Acc.	Unk. Acc.
3GRAMMEMM	See text	248,798	52.07%	96.92%	88.99%
NAACL 2003	See text and [1]	460,552	55.31%	97.15%	88.61%
Replication	See text and [1]	460,551	55.62%	97.18%	88.92%
Replication′	+rareFeatureThresh = 5	482,364	55.67%	97.19%	88.96%
5W	$+\langle t_0, w_{-2}\rangle, \langle t_0, w_2\rangle$	730,178	56.23%	97.20%	89.03%
5WSHAPES	$+\langle t_0, s_{-1}\rangle, \langle t_0, s_0\rangle, \langle t_0, s_{+1}\rangle$	731,661	56.52%	97.25%	89.81%
5WSHAPESDS	+ distributional similarity	737,955	56.79%	97.28%	90.46%



Feature Engineering for NLP

Suppose you want to predict whether the word is the *root* (i.e. predicate) of the sentence.

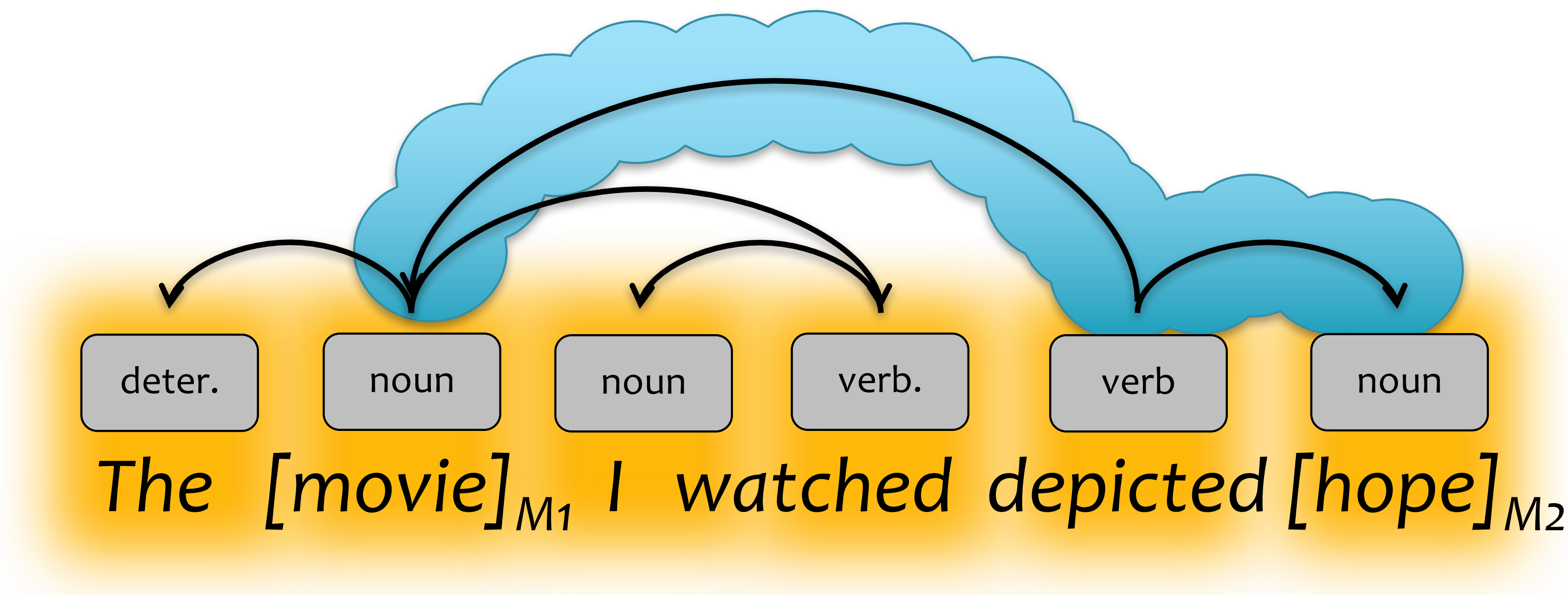
What features should you use?



Feature Engineering for NLP

Per-word Features:

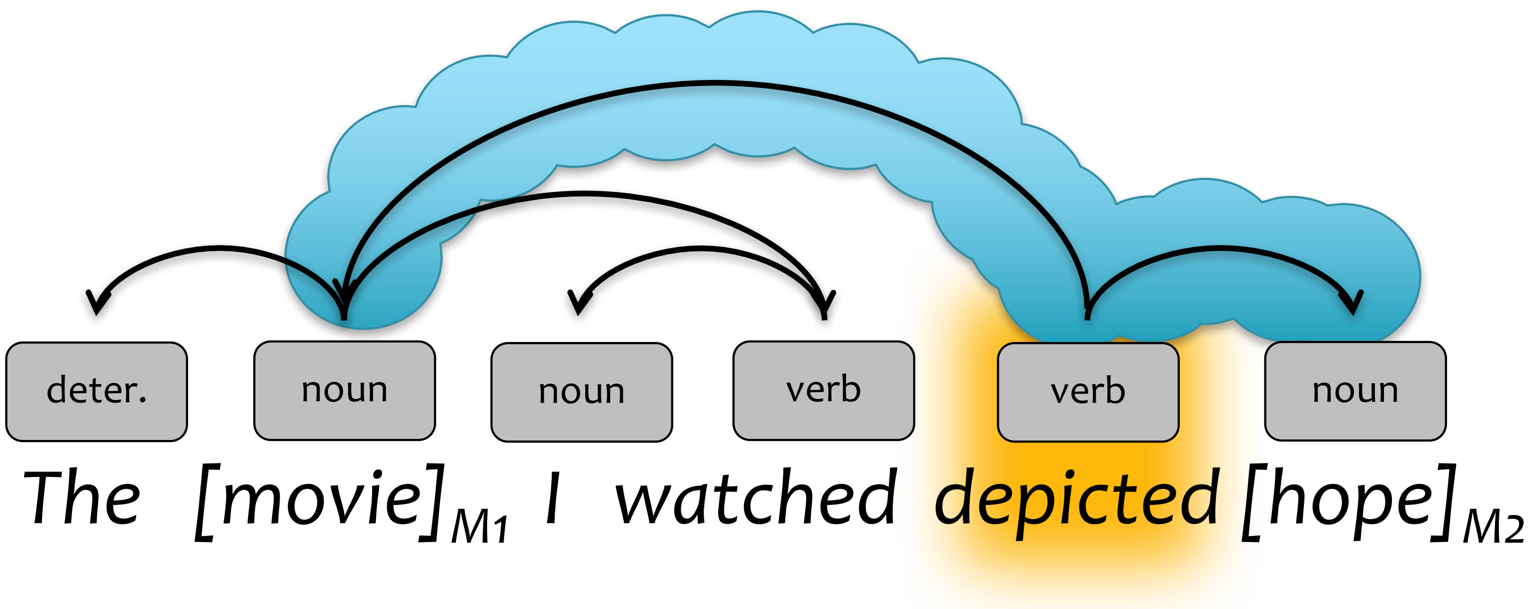
	f_1	f_2	f_3	f_4	f_5	f_6
`on-path(w_i)`	0	1	0	0	1	1
`is-between(w_i)`	0	0	1	1	1	0
`head-of-M1(w_i)`	0	1	0	0	0	0
`head-of-M2(w_i)`	0	0	0	0	0	1
`before-M1(w_i)`	1	0	0	0	0	0
`before-M2(w_i)`	0	0	0	0	1	0
…	…	…	…	…	…	…



Feature Engineering for NLP

Per-word Features:

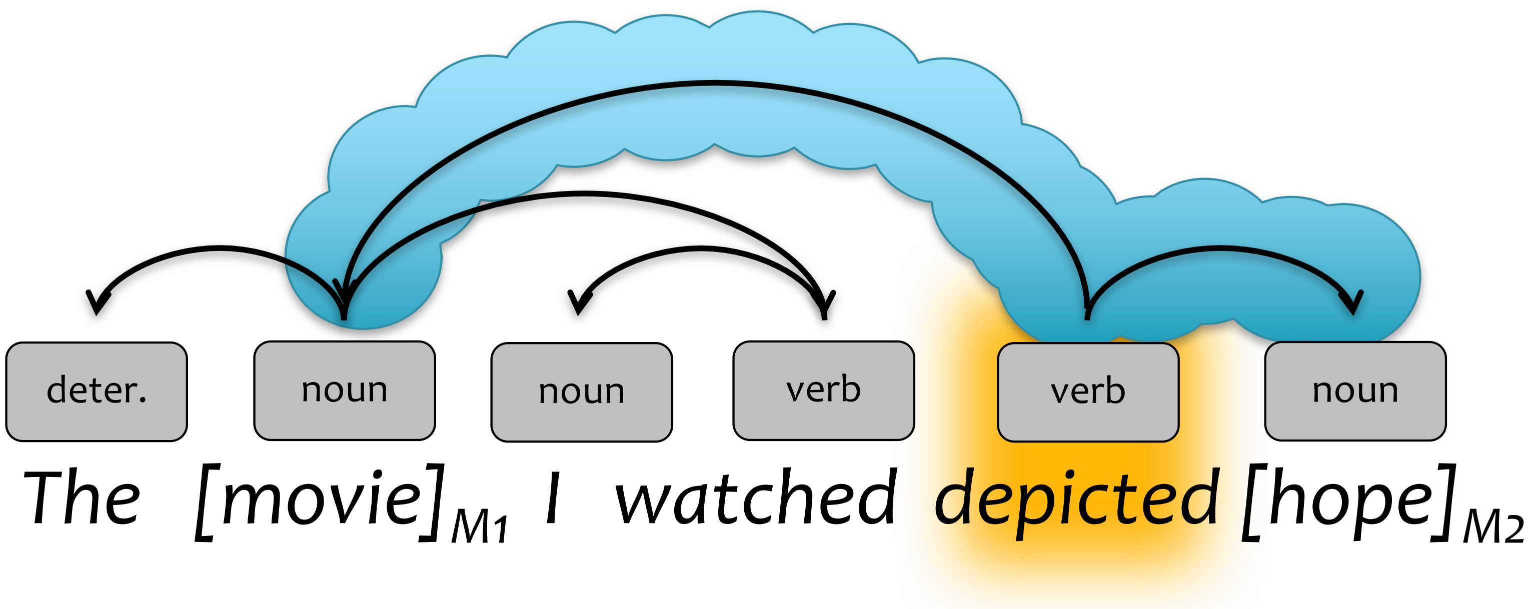
	f_5
`on-path(`w_i`)`	1
`is-between(`w_i`)`	1
`head-of-M1(`w_i`)`	0
`head-of-M2(`w_i`)`	0
`before-M1(`w_i`)`	0
`before-M2(`w_i`)`	1
…	…



Feature Engineering for NLP

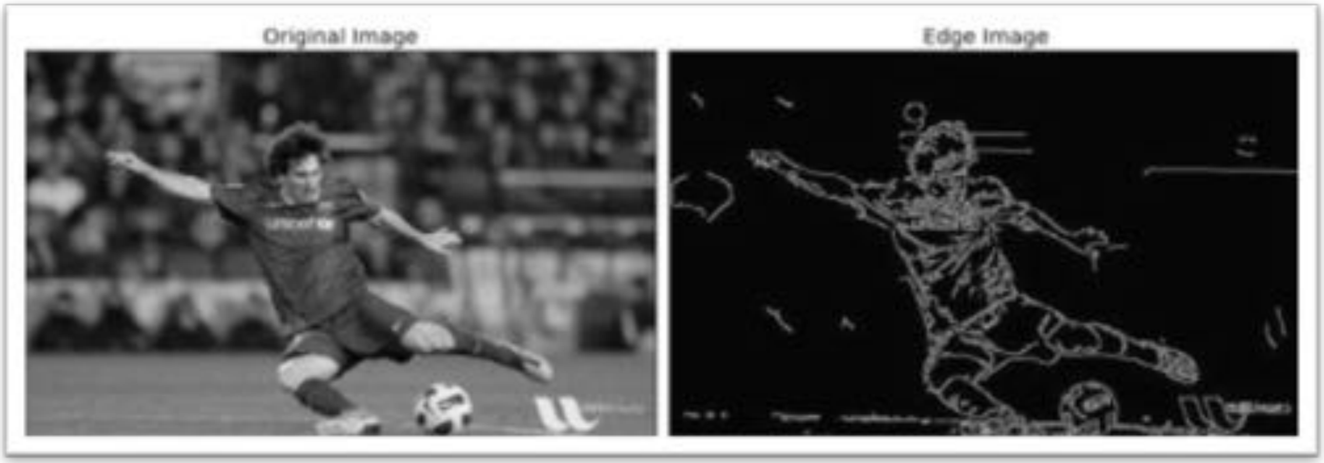
Per-word Features: (with conjunction)

	f_5
`on-path(`w_i`) && `w_i`== "depicted"`	1
`is-between(`w_i`) && `w_i`== "depicted"`	1
`head-of-M1(`w_i`) && `w_i`== "depicted"`	0
`head-of-M2(`w_i`) && `w_i`== "depicted"`	0
`before-M1(`w_i`) && `w_i`== "depicted"`	0
`before-M2(`w_i`) && `w_i`== "depicted"`	1
…	…

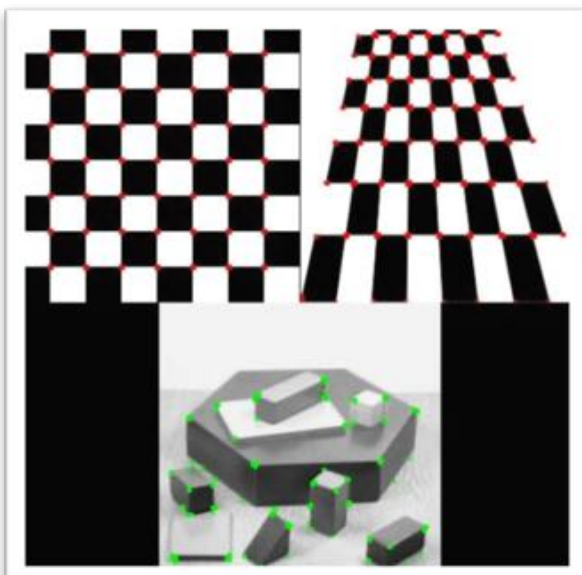


Feature Engineering for CV

Edge detection (Canny)



Corner Detection (Harris)



Figures from http://opencv.org

Feature Engineering for CV

Scale Invariant Feature Transform (SIFT)

Figure 3: Model images of planar objects are shown in the top row. Recognition results below show model outlines and image keys used for matching.

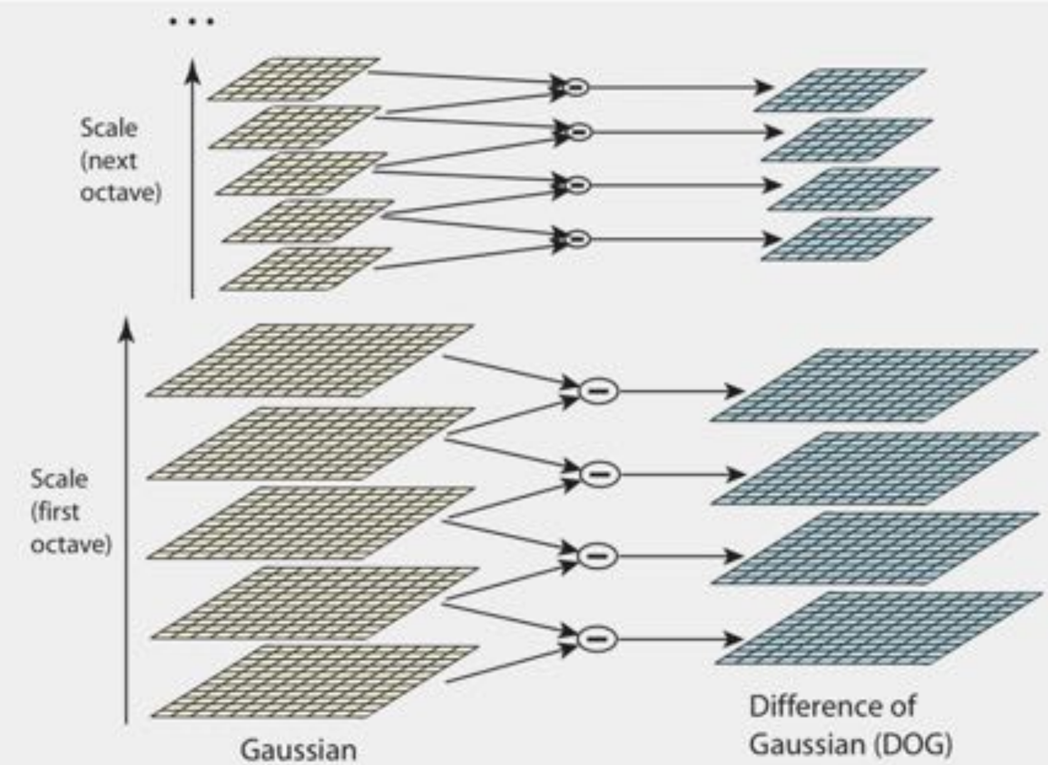


Figure 1: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated.

Figure from Lowe (1999) and Lowe (2004)