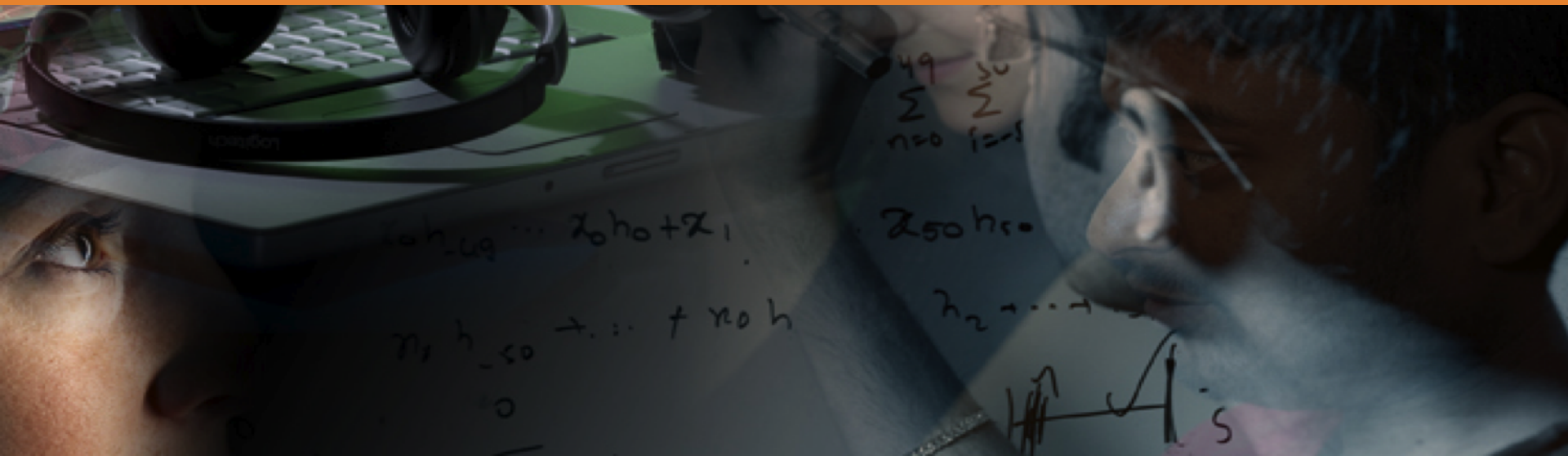
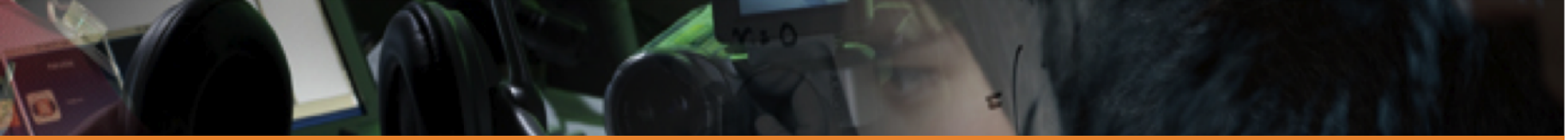
Natural Language Processing (Almost) from Scratch

Ronan Collobert

International Advisory Board Meeting, Idiap, Sep. 2, 2011
Research Activities



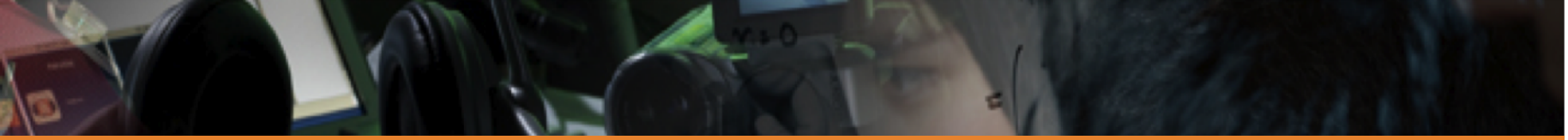


Goal

How to convert a piece of English text
into a programmer friendly data structure
that describes the meaning of the natural language text?

- Natural Language Processing benchmarks
  = indirect measurements of representation relevance

POS	Part-Of-Speech
CHK	Chunking
NER	Name Entity Recognition
SRL	Semantic Role Labeling
PSG	Syntactic Parsing



Part I

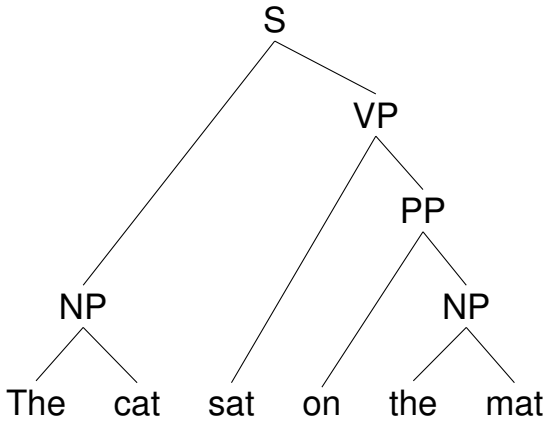
The Background

Natural Language Processing Tasks

- Part-Of-Speech Tagging (POS): syntactic roles (noun, adverb...)
- Chunking (CHK): syntactic constituents (noun phrase, verb phrase...)
- Name Entity Recognition (NER): person/company/location...
- Semantic Role Labeling (SRL): semantic role

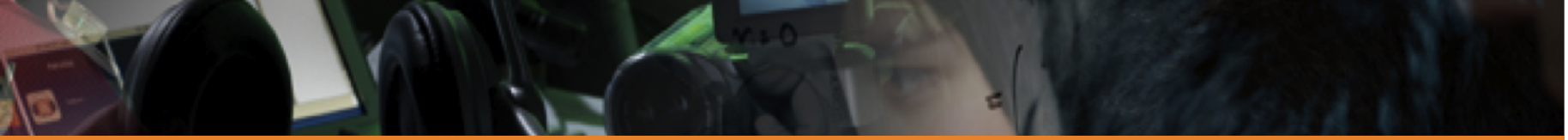
  [John]$_{ARG0}$ [ate]$_{REL}$ [the apple]$_{ARG1}$ [in the garden]$_{ARGM-LOC}$

- Parsing (PSG):



- Tagging tasks (BIOES tagging scheme):

The	black	cat	sat	on	the	mat	.
B-NP	I-NP	E-NP	S-VP	S-PP	B-NP	E-NP	O



NLP Benchmarks

- Datasets:
  - ⋆ POS, CHK, SRL: WSJ (≈ up to 1M labeled words)
  - ⋆ NER: Reuters (≈ 200K labeled words)

System	Accuracy
Shen, 2007	97.33%
Toutanova, 2003	**97.24%**
Gimenez, 2004	97.16%

(a) POS: As in (Toutanova, 2003)

System	F1
Shen, 2005	95.23%
Sha, 2003	**94.29%**
Kudoh, 2001	93.91%

(b) CHK: CoNLL 2000

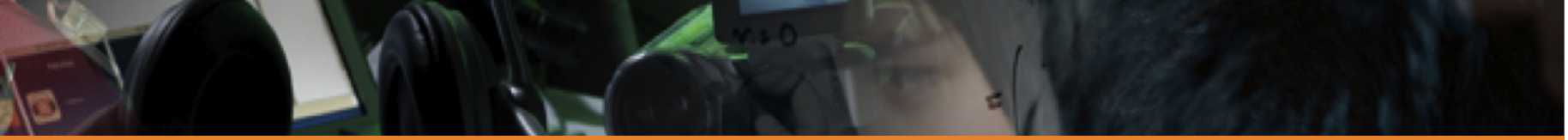
System	F1
Ando, 2005	**89.31%**
Florian, 2003	88.76%
Kudoh, 2001	88.31%

(c) NER: CoNLL 2003

System	F1
Koomen, 2005	**77.92%**
Pradhan, 2005	77.30%
Haghighi, 2005	77.04%

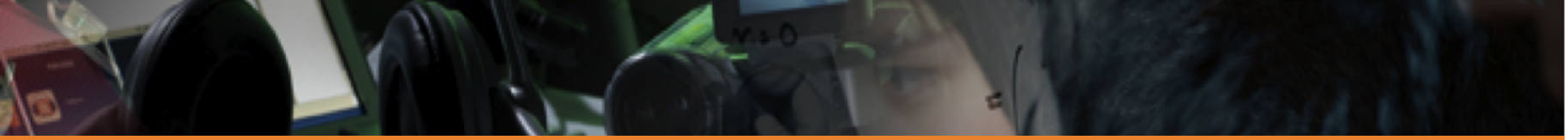
(d) SRL: CoNLL 2005

- We chose as benchmark systems:
  - ⋆ Well-established systems
  - ⋆ Systems avoiding external labeled data



Standard NLP Benchmarks

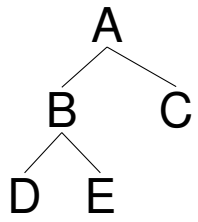
POS (Toutanova, 2003)	Various combinations of surrounding words & tags, various caps, digit, dash, various prefixes & suffixes Dependency Network
Chunking (Sha, 2003)	Surrounding words, POS tags Conditional Random Field (CRF)
NER (Ando, 2005)	Surrounding words, POS, several suffixes & prefixes, surrounding tags, bigrams, previously assigned tags to words, unlabeled data Viterbi decoding at test
SRL (Koomen, 2005)	6 parse trees, pruning heuristics, POS, voice, phrase type, head words, subparts of the trees, ... Argument identification, argument classification, integer linear programming



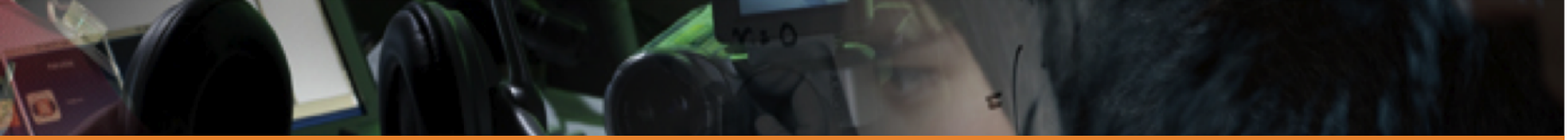
Standard NLP Benchmarks

PCFG

$A \longrightarrow B\ C$

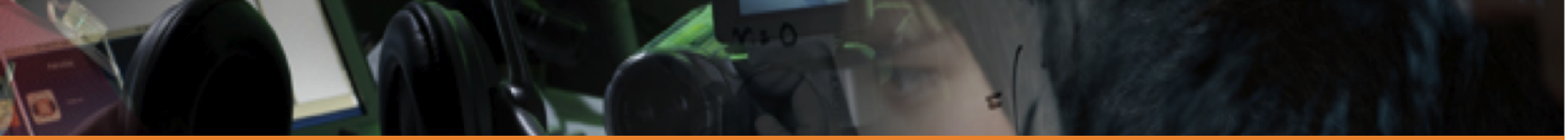


Parsing (Collins, 1999) (Charniak, 2000)	Lexicalized Probabilistic Context-Free Grammar (PCFG), POS, head words, chart parser, deleted interpolation, ... 30 pages of details in (Bikel, 2004)
Parsing (Charniak & Johnson, 2005 & 2006)	Re-ranking over the above, using lots of ad-hoc features
Parsing (Finkel et al, 2008) (Petrov & Klein, 2008) (Carreras & al, 2008	PCFG, dependency features CRF or similar



Machine Learning

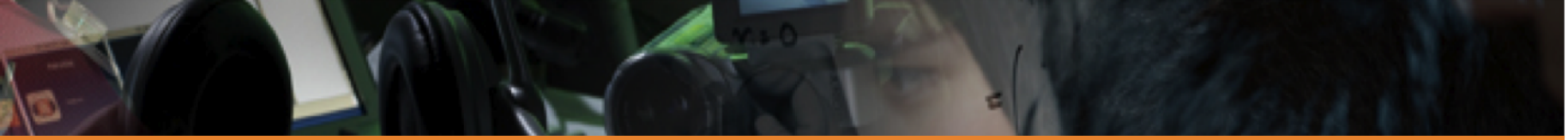
In “Machine Learning”
There is “Learning”



Machine Learning

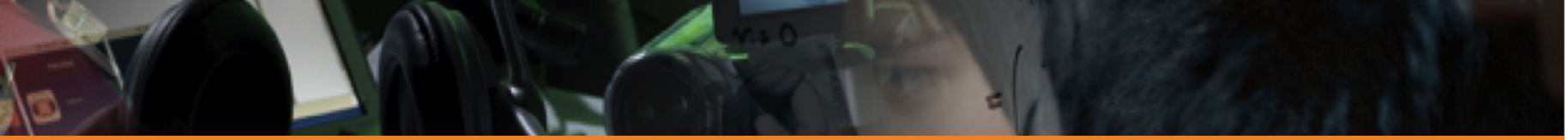
Trade task specific engineering for more generic (complex?) model which learns features for you.

complex features	$\longrightarrow$	simple features
linear model	$\longrightarrow$	non-linear neural network
CRF	$\longrightarrow$	Graph Transformer Network
		see (Bottou et al, 1991)



Part II

The Networks



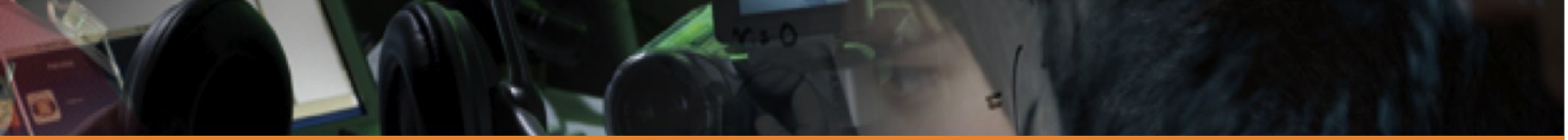
Neural Networks

Stack several layers together (increase level of abstraction)
Parameters M^i trained by gradient descent

x	input (vector)
$M^1 \times \cdot$	linear operation (embedding)
$\tanh(\cdot)$	non-linearity
$M^2 \times \cdot$	linear operation (embedding)
	score per label

Convolutional layer

$X = (X_{\bullet 1},\ X_{\bullet 2} \cdots)$	input (matrix)
$M \times \begin{pmatrix} X_{\bullet 1} & X_{\bullet 2} & \\ X_{\bullet 2} & X_{\bullet 3} & \cdots \\ X_{\bullet 3} & X_{\bullet 4} & \end{pmatrix}$	convolution (local embedding for each input column)



Words into Vectors

$$\begin{array}{rcl} \text{a word} & = & \text{index in a dictionary} \\ \text{The cat sat on the mat} & = & (w_1,\, w_2,\, w_3,\, w_4,\, w_5,\, w_6) \end{array}$$

binary code $\sim$ dictionary size

$$w \longleftrightarrow \Big(0, \cdots 0, \underset{\text{at index } w}{1}, 0, \cdots 0\Big)^{\mathrm{T}} = (\mathbf{1}_{\cdot=w})^{\mathrm{T}}$$

word embedding

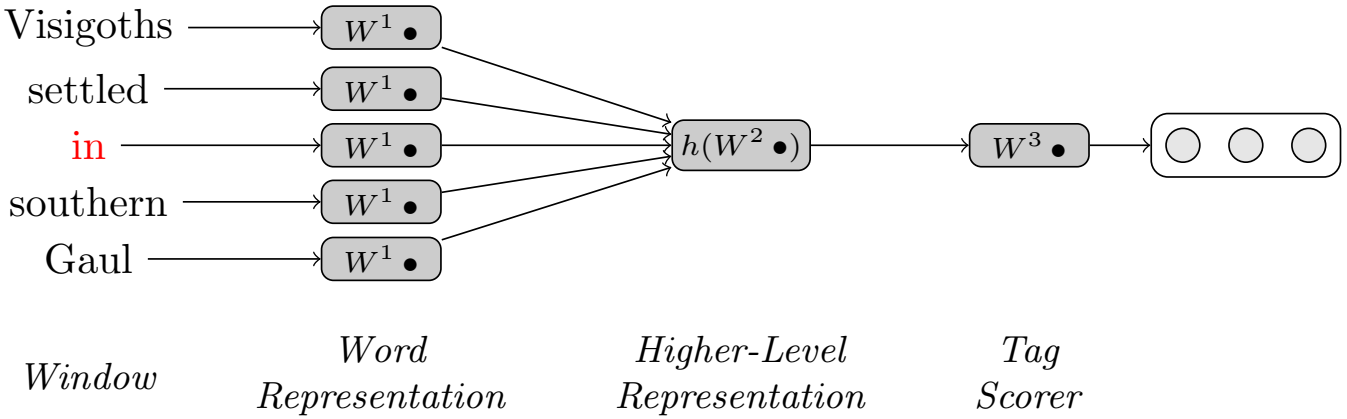
$M \sim$ feature size $\times$ dictionary size

$$M \times (\mathbf{1}_{\cdot=w}) = M_{\bullet w}$$

lookup-table operation

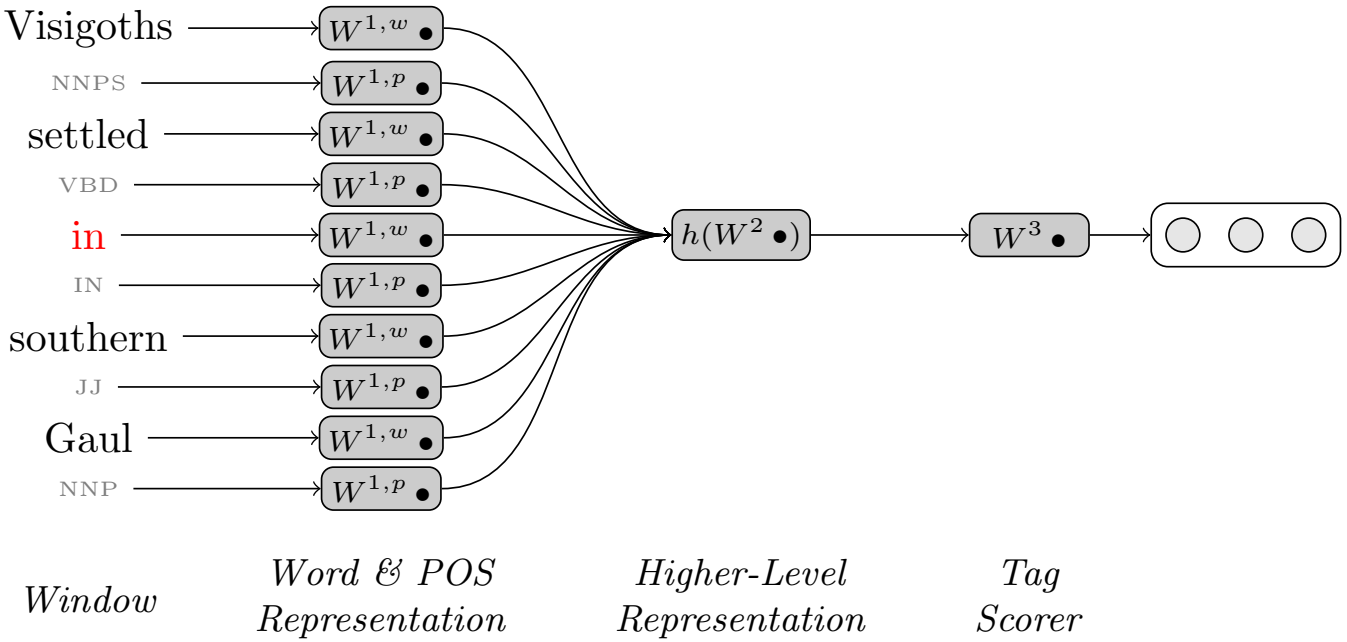
Window Approach

How to tag “in” in the sentence
“The Visigoths settled in southern Gaul”?



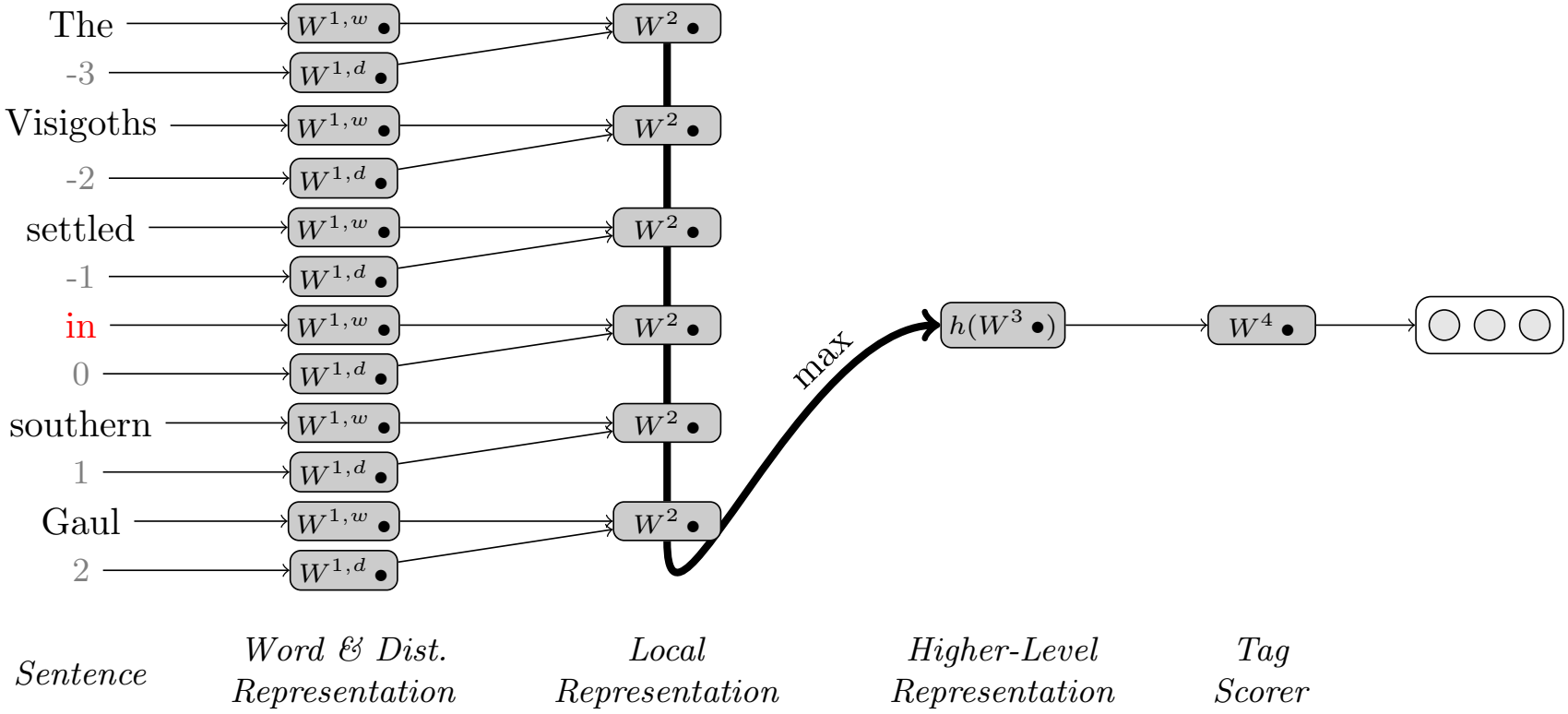
Window Approach (Extra Features)

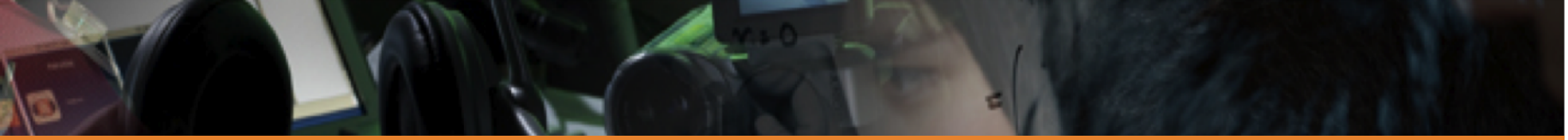
How to tag “in” in the sentence
“The Visigoths settled in southern Gaul”?



Sentence Approach

How to tag “in” in the sentence
“The Visigoths settled in southern Gaul”?





Training

- Given a training set $\mathcal{T}$
- Convert network outputs into probabilities
- Maximize a log-likelihood

$$\boldsymbol{\theta} \longmapsto \sum_{(\boldsymbol{x}, y) \in \mathcal{T}} \log p(y \,|\, \boldsymbol{x}, \boldsymbol{\theta})$$

- Use stochastic gradient ascent

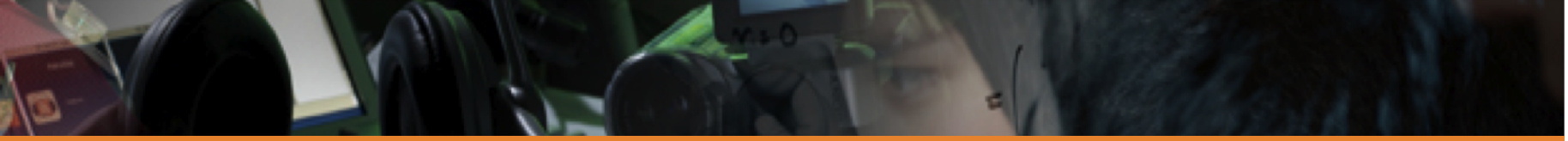
$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} + \lambda \frac{\partial \log p(y \,|\, \boldsymbol{x}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

Fixed learning rate. "Tricks":

- ⋆ Divide learning rate by "fan-in"
- ⋆ Initialization according to "fan-in"

- Use chain rule ("back-propagation") for efficient gradient computation

- How to interpret neural networks outputs as probabilities?



Word-Level Likelihood (WLL)

- The network has one output $f(\boldsymbol{x}, i, \boldsymbol{\theta})$ per tag i
- Interpreted as a probability with a softmax over all tags

$$p(i \mid \boldsymbol{x}, \boldsymbol{\theta}) = \frac{e^{f(\boldsymbol{x}, i, \boldsymbol{\theta})}}{\sum_j e^{f(\boldsymbol{x}, j, \boldsymbol{\theta})}}$$

- Define the logadd operation

$$\operatorname*{logadd}_i z_i = \log\left(\sum_i e^{z_i}\right)$$

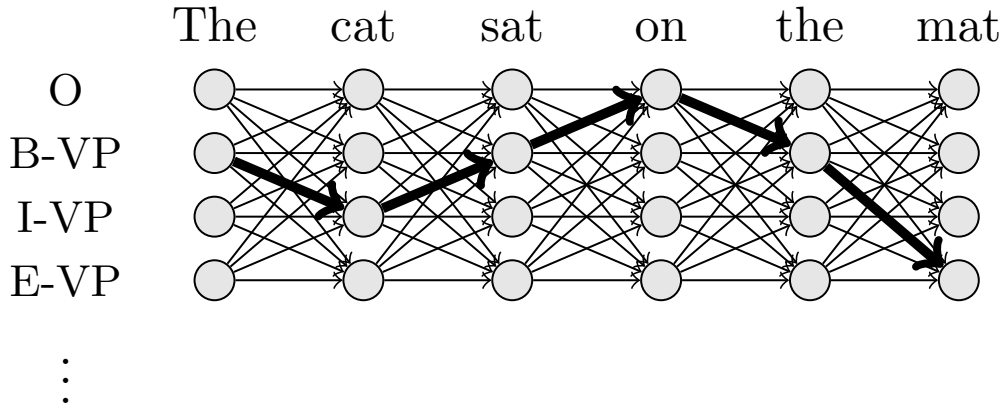
- Log-likelihood for example $(\boldsymbol{x}, y)$

$$\log p(y \mid \boldsymbol{x}, \boldsymbol{\theta}) = f(\boldsymbol{x}, y, \boldsymbol{\theta}) - \operatorname*{logadd}_j f(\boldsymbol{x}, j, \boldsymbol{\theta})$$

- How to leverage the sentence structure?

Sentence-Level Likelihood (SLL) (1/2)

- The network score for tag k at the t^{th} word is $f([\boldsymbol{x}]_1^T, k, t, \boldsymbol{\theta})$
- A_{kl} transition score to jump from tag k to tag l



- Sentence score for a tag path $[i]_1^T$

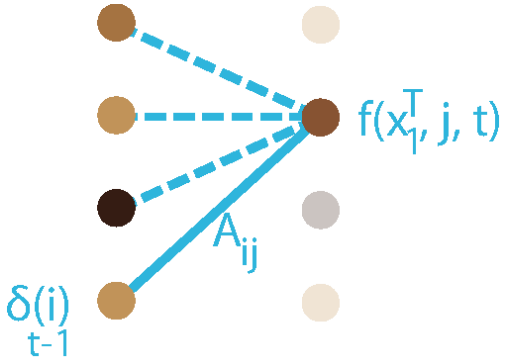
$$s([\boldsymbol{x}]_1^T, [i]_1^T, \tilde{\boldsymbol{\theta}}) = \sum_{t=1}^{T} \left(A_{[i]_{t-1}[i]_t} + f([\boldsymbol{x}]_1^T, [i]_t, t, \boldsymbol{\theta}) \right)$$

- Conditional likelihood by normalizing w.r.t all possible paths:

$$\log p([y]_1^T \mid [\boldsymbol{x}]_1^T, \tilde{\boldsymbol{\theta}}) = s([\boldsymbol{x}]_1^T, [y]_1^T, \tilde{\boldsymbol{\theta}}) - \underset{\forall [j]_1^T}{\text{logadd}}\, s([\boldsymbol{x}]_1^T, [j]_1^T, \tilde{\boldsymbol{\theta}})$$

Sentence-Level Likelihood (SLL) (2/2)

- Normalization computed with recursive Forward algorithm:



$$\delta_t(j) = \text{logAdd}_i \left[\delta_{t-1}(i) + A_{i,j} + f_\theta(j, x_1^T, t)\right]$$

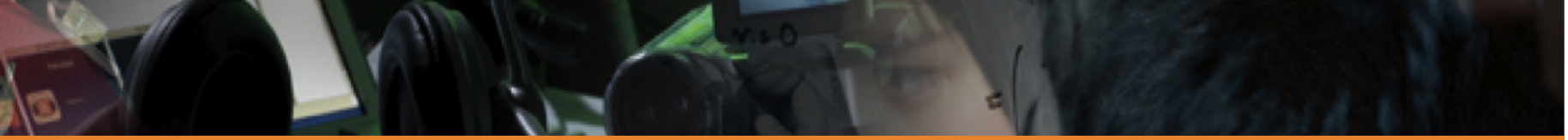
Termination:

$$\underset{\forall [j]_1^T}{\text{logadd}}\, s([\boldsymbol{x}]_1^T, [j]_1^T, \tilde{\boldsymbol{\theta}}) = \text{logAdd}_i\, \delta_T(i)$$

- Simply backpropagate through this recursion with chain rule

- Non-linear CRFs: Graph Transformer Networks (Bottou, 1997)
- Compared to CRFs, we train features (network parameters $\boldsymbol{\theta}$ and transitions scores A_{kl})

- Inference: Viterbi algorithm (replace logAdd by max)

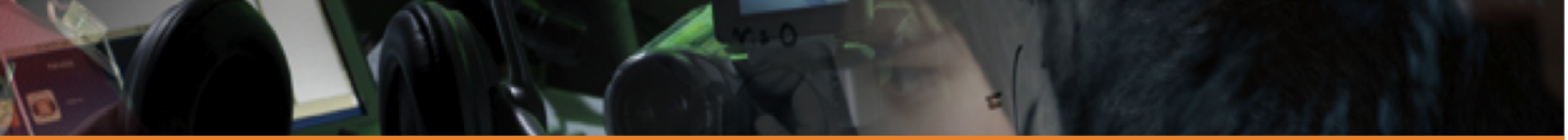


Supervised Benchmark Results

- Network architectures:
  - ⋆ Window (5) approach for POS, CHK & NER (300HU)
  - ⋆ Sentence approach for SRL (300+500HU)
  - ⋆ Word-Level Likelihood (WLL) and Sentence-Level Likelihood (SLL)
- Network features: lower case words (size 50), capital letters (size 5)
  dictionary size 100,000 words

Approach	**POS** (PWA)	**Chunking** (F1)	**NER** (F1)	**SRL** (F1)
Benchmark Systems	**97.24**	**94.29**	**89.31**	**77.92**
NN+WLL	96.31	89.13	79.53	54.53
NN+SLL	96.37	90.33	81.47	71.24

- SLL helps, but... fair performance.
- Capacity mainly in words embeddings... are we training them right?

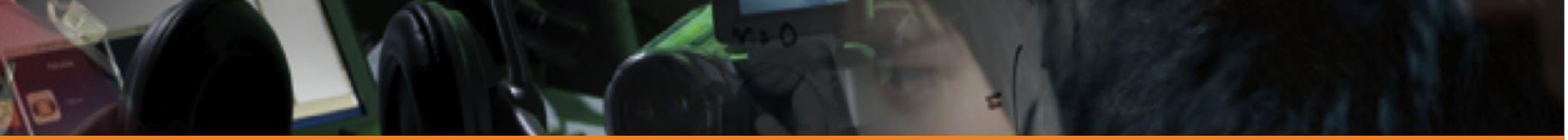


Supervised Word Embeddings

- Sentences with similar words should be tagged in the same way:
  - ⋆ The cat sat on the mat
  - ⋆ The feline sat on the mat

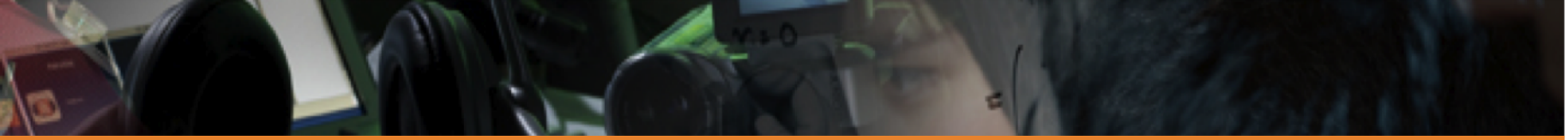
FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
454	1973	6909	11724	29869	87025
PERSUADE	THICKETS	DECADENT	WIDESCREEN	ODD	PPA
FAW	SAVARY	DIVO	ANTICA	ANCHIETA	UDDIN
BLACKSTOCK	SYMPATHETIC	VERUS	SHABBY	EMIGRATION	BIOLOGICALLY
GIORGI	JFK	OXIDE	AWE	MARKING	KAYAK
SHAHEED	KHWARAZM	URBINA	THUD	HEUER	MCLARENS
RUMELIA	STATIONERY	EPOS	OCCUPANT	SAMBHAJI	GLADWIN
PLANUM	ILIAS	EGLINTON	REVISED	WORSHIPPERS	CENTRALLY
GOA'ULD	GSNUMBER	EDGING	LEAVENED	RITSUKO	INDONESIA
COLLATION	OPERATOR	FRG	PANDIONIDAE	LIFELESS	MONEO
BACHA	W.J.	NAMSOS	SHIRT	MAHAN	NILGIRIS

- About 1M of words in WSJ
- 15% of most frequent words in the dictionary are seen 90% of the time
- Cannot expect words to be trained properly!



Part III

Lots Of Unlabeled Data



Ranking Language Model

- Language Model: "*is a sentence actually english or not?*"
  Implicitly captures: ⋆ syntax ⋆ semantics
- Bengio & Ducharme (2001) Probability of next word given previous words.
  Overcomplicated – we do not need probabilities here
- $f()$ a window approach network
- Ranking margin cost:

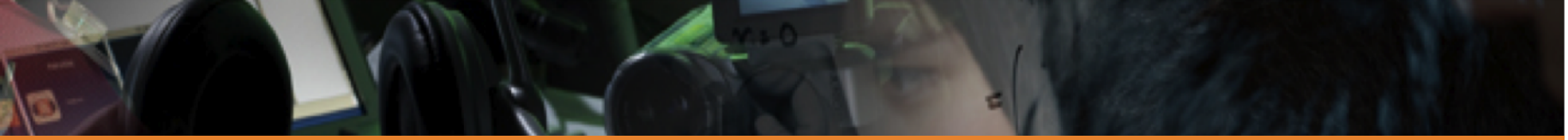
$$\sum_{s \in \mathcal{S}} \sum_{w \in \mathcal{D}} \max(0,\, 1 - f(s,\, w_s^\star) + f(s,\, w))$$

$\mathcal{S}$: sentence windows $\mathcal{D}$: dictionary

$w_s^\star$: true middle word in s

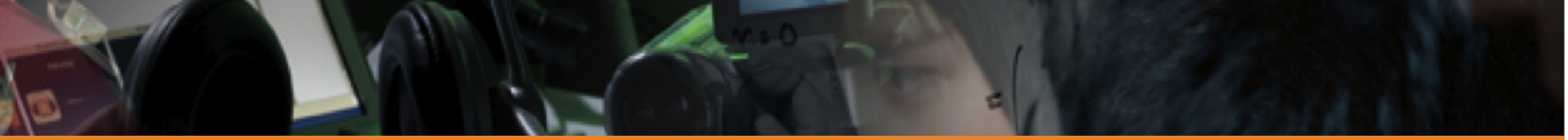
$f(s,\, w)$: network score for sentence s and middle word w

- Stochastic training:
  - ⋆ Positive example: random corpus sentence
  - ⋆ Negative example: replace middle word by random word



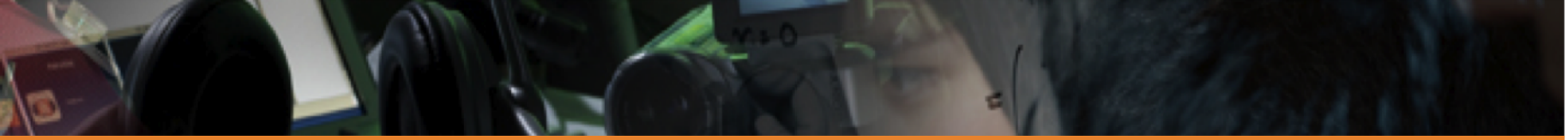
Training Language Model

- Two window approach (11) networks (100HU) trained on two corpus:
  - ⋆ LM1: Wikipedia: **631M** of words
  - ⋆ LM2: Wikipedia+Reuters RCV1: **631M+221M=852M** of words
- LM1
  - ⋆ order dictionary words by frequency
  - ⋆ increase dictionary size: $5000, 10,000, 30,000, 50,000, 100,000$
  - ⋆ 4 weeks of training
- LM2
  - ⋆ initialized with LM1, dictionary size is $130,000$
  - ⋆ 30,000 additional most frequent Reuters words
  - ⋆ 3 additional weeks of training



Unsupervised Word Embeddings

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
454	1973	6909	11724	29869	87025
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

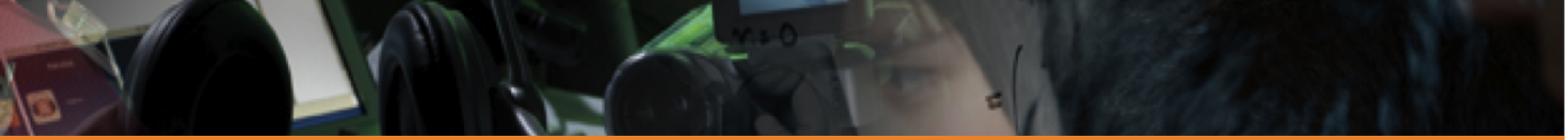


Semi-Supervised Benchmark Results

- Initialize word embeddings with LM1 or LM2
- Same training procedure

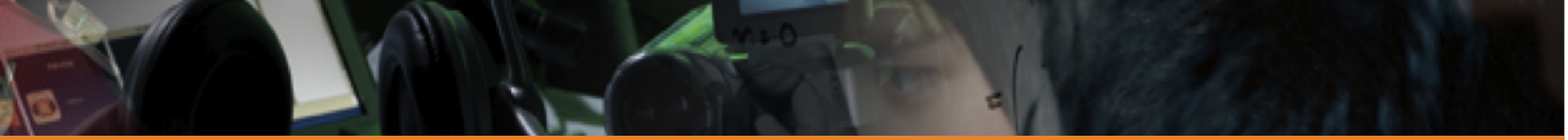
Approach	POS (PWA)	CHK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	**97.24**	**94.29**	**89.31**	**77.92**
NN+WLL	96.31	89.13	79.53	54.53
NN+SLL	96.37	90.33	81.47	71.24
NN+WLL+LM1	97.05	91.91	85.68	57.32
NN+SLL+LM1	97.10	93.65	87.58	74.28
NN+WLL+LM2	97.14	92.04	86.96	56.97
NN+SLL+LM2	97.20	93.63	88.67	73.90

- Huge boost from language models



Part IV

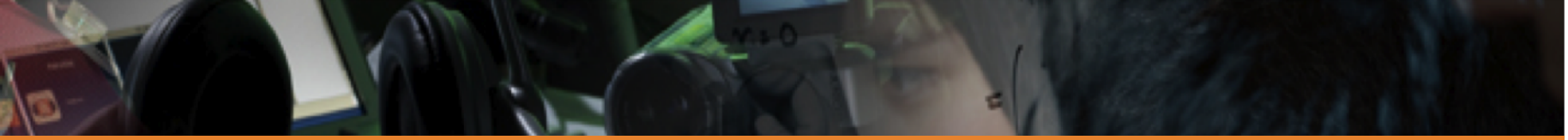
The Temptation



Cascading Tasks

Increase level of engineering by incorporating common NLP techniques

- Stemming for western languages benefits POS (Ratnaparkhi, 1996)
  - ⋆ Use last two characters as feature (455 different stems)
- Gazetteers are often used for NER (Florian, 2003)
  - ⋆ $8,000$ locations, person names, organizations and misc entries from CoNLL 2003
- POS is a good feature for CHK & NER (Shen, 2005) (Florian, 2003)
  - ⋆ We feed our own POS tags as feature
- CHK is also a common feature for SRL (Koomen, 2005)
  - ⋆ We feed our own CHK tags as feature



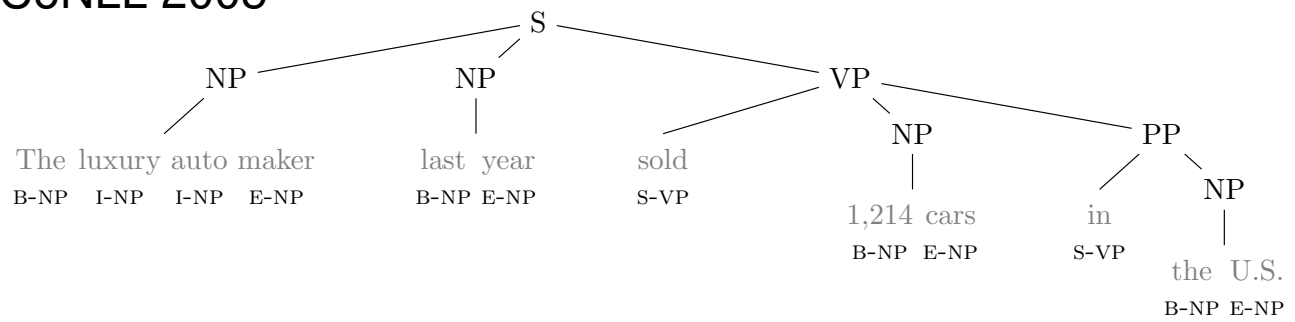
Cascading Tasks Benchmark Results

Approach	POS (PWA)	CHK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	**97.24**	**94.29**	**89.31**	**77.92**
NN+SLL+LM2	97.20	93.63	88.67	73.90
NN+SLL+LM2+Suffix2	97.29	–	–	–
NN+SLL+LM2+Gazetteer	–	–	89.59	–
NN+SLL+LM2+POS	–	94.32	88.67	75.39
NN+SLL+LM2+CHK	–	–	–	74.73

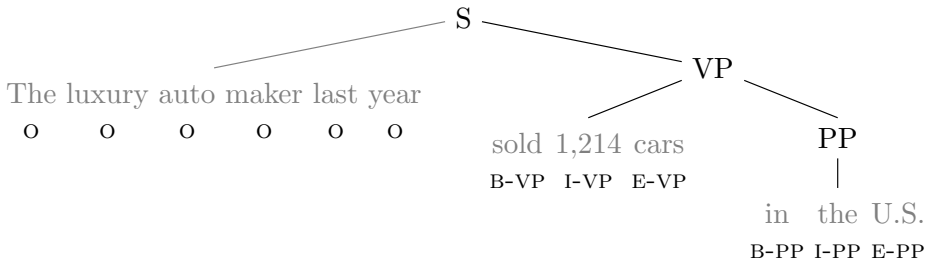
Parsing for SRL

- Parsing is essential to SRL (Punyakanok, 2005) (Pradhan, 2005)
- State-of-the-art SRL systems use several parse trees (up to 6!!)
- We feed our network several levels of Charniak parse tree provided by CoNLL 2005

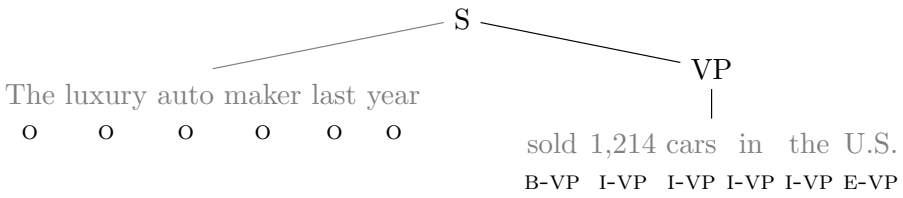
LEVEL 1

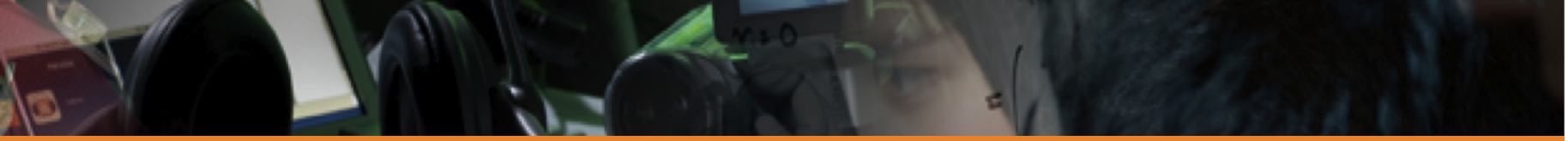


LEVEL 2



LEVEL 3

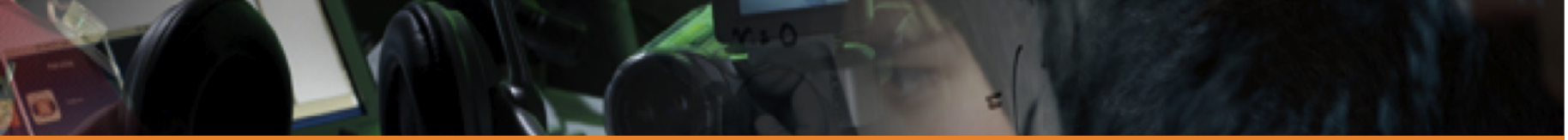




SRL Benchmark Results With Parsing

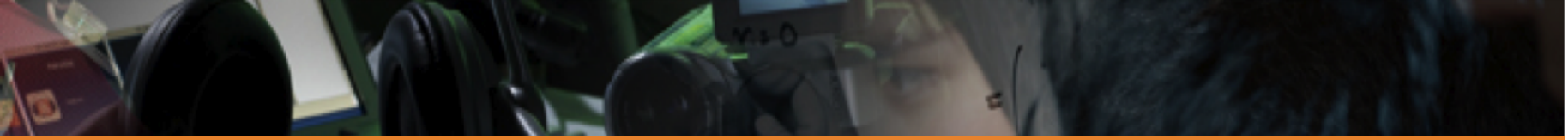
Approach	SRL (test set F1)
Benchmark System (six parse trees)	**77.92**
Benchmark System (top Charniak only)	**74.76**†
NN+SLL+LM2	73.90
NN+SLL+LM2+CHK	74.73
NN+SLL+LM2+Charniak (level 1 only)	76.27
NN+SLL+LM2+Charniak (levels 1 & 2)	76.24
NN+SLL+LM2+Charniak (levels 1 to 3)	76.62
NN+SLL+LM2+Charniak (levels 1 to 4)	76.50
NN+SLL+LM2+Charniak (levels 1 to 5)	76.98

† on the validation set



Part V

Implementation



SENNA

- Implements our networks in simple C ($\approx$ 3000 lines)
- Includes POS, CHK, NER, SRL and PSG tasks!

Available at `http://ml.nec-labs.com/software/senna`

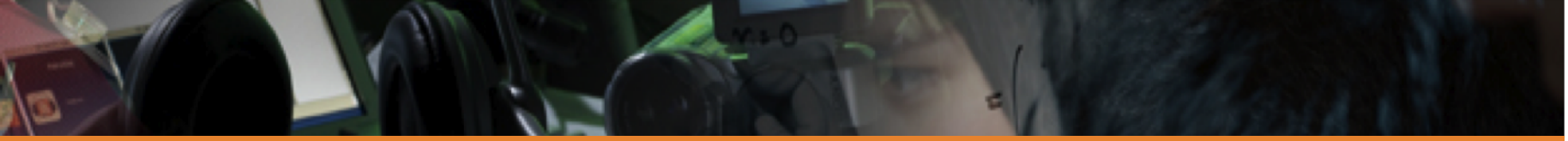
- Neural network = stack of matrix-vector multiplications $\longrightarrow$ use BLAS!
- Order of magnitude faster than existing systems

System	RAM (Mb)	Time (s)
Toutanova, 2003	1100	1065
Shen, 2007	2200	833
NN	32	4

(a) POS

System	RAM (Mb)	Time (s)
Koomen, 2005	3400	6253
NN	124	52

(b) SRL



Conclusion

Achievements

- "All purpose" neural network architecture for NLP
- Limit task-specific engineering
- Rely on very large unlabeled datasets: generic word features
- We do not plan to stop here

Critics

- Why forgetting NLP expertise for neural network training skills?
  - ⋆ NLP goals are not limited to existing NLP task
  - ⋆ Excessive task-specific engineering is not desirable
- Why neural networks?
  - ⋆ Scale on massive datasets
  - ⋆ Discover hidden representations
  - ⋆ Most of neural network technology existed in 1997

If we had started in 1997 with vintage computers,
training would be near completion today!!