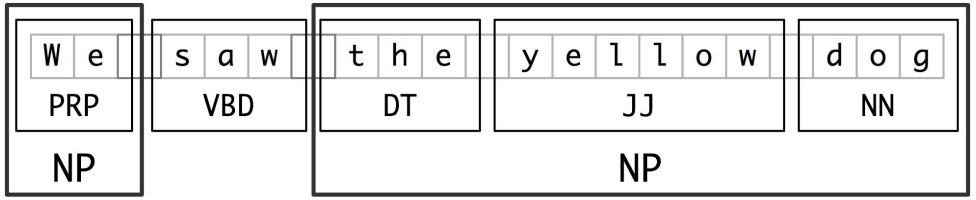
Natural Language Processing (Almost) from Scratch

Ronan Collobert et al., J. of Machine Learning Research, 2011

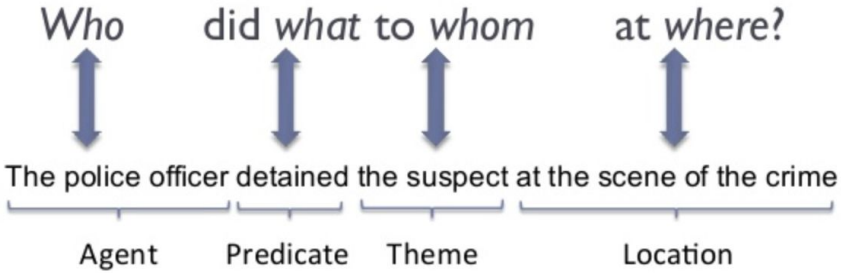
Guanqun Yang, Pencheng Xu,
Haiqi Xiao, Yuqing Wang

2. Benchmark tasks

- Sequence labeling task
- Assign discrete labels to discrete elements in a sequence
  - Part-of-speech tagging (POS)
  - Chunking (CHUNK)
  - Named entity recognition (NER)
  - Semantic role labeling (SRL)



Automatically find names of people, places, products, and organizations in text across many languages.

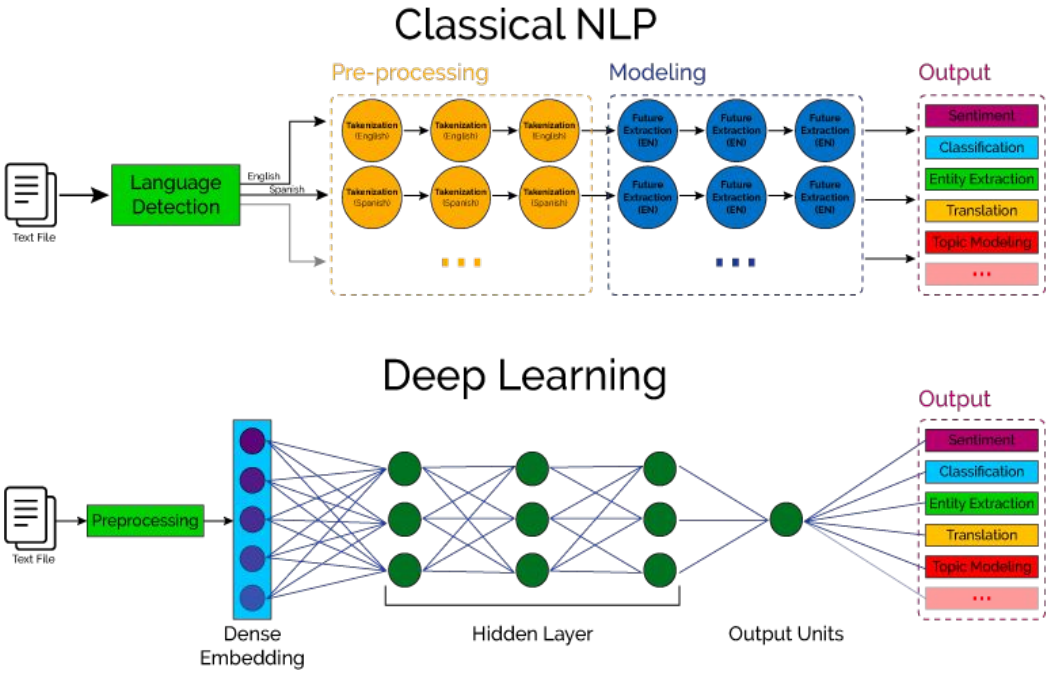


Overview

1. Introduction
2. Benchmark tasks
3. Neural network architecture
4. Techniques to improve the performance of NN
    - Unlabeled data
    - Multi-task learning
    - Task-specific engineering

1. Introduction

- Many NLP tasks involve assigning labels in words
- “Almost from scratch”
- Excel on multiple benchmarks
- Avoid task-specific engineering
- “Use a single learning system able to discover adequate internal representations”



2.1 POS Tagging

- Assign tag to each word in the sentence to represent its grammatical role
  - A simplistic version may involve Verb, Noun, Determiner, etc
  - Could be more fine-grained in some systems.
- Previously solved with traditional machine learning algorithms
  - Averaged perceptron algorithm (used by NLTK)
  - HMM, CRF (previously covered in class)
- *Prone to mistakes* when syntactic and semantic ambiguity appears.

- Example
  - Can of fish.
  - They can fish.

```
nltk.pos_tag(["can", "of", "fish"])
[('can', 'MD'), ('of', 'IN'), ('fish', 'NN')]

nltk.pos_tag(["they", "can", "fish"])
[('they', 'PRP'), ('can', 'MD'), ('fish', 'VB')]
```

- Example
  - The old man the boat.

```
nltk.pos_tag(["the", "old", "man", "the", "boat"])
[('the', 'DT'), ('old', 'JJ'), ('man', 'NN'), ('the', 'DT'), ('boat', 'NN')]
```

2.2 Chunking (CHUNK)

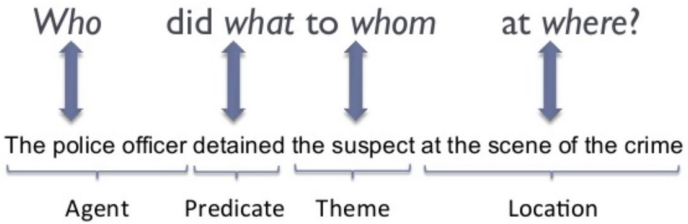
- Defining a **Chunk grammar**
- Creating a **Chunk parser**
- Output will be a **tree** which shows all chunks presents in that sentence
- An example of Noun-Phrase Chunker
- Sentence = [(“a”, “DT”), (“beautiful”, “JJ”), (“young”, “JJ”), (“lady”, “NN”), (“is”, ”VBP”), (“walking”, “VBP”), (“on”, “IN”), (“the”, “DT”), (“pavement”, “NN”),]
- Grammar = “NP: {<DT>?<JJ>*<NN>}”
- What if Grammar is “NP: {<DT>?<JJ>+<NN>}”

2.3 Named Entity Recognition

- Locate and classify named entity in unstructured text into predefined categories such as person names, organizations, locations, times and so on.
- Example:
    - Jim bought 300 shares of Acme Corp. in 2006.
    - [Jim]Person bought 300 shares of [Acme Corp.]Organization in [2006]Time.

2.4 Semantic Role Labeling

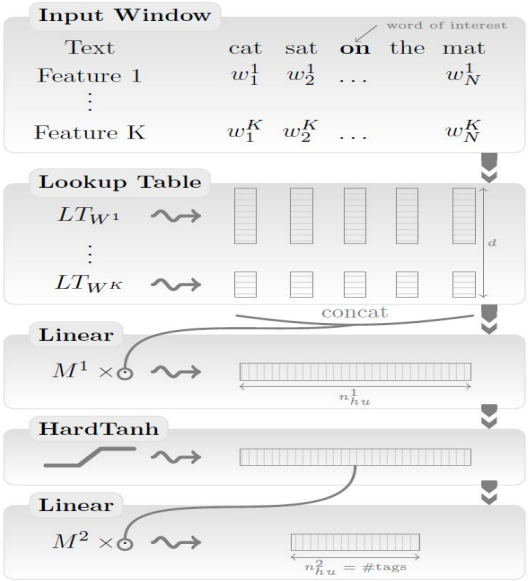
- Given a predicate (verb phrases),
  label the semantic roles of words in the same sentence.
  - core arguments: agent/patient/theme/experiencer/beneficiary...
  - adjunctive arguments: time/location/direction/degree/frequency...



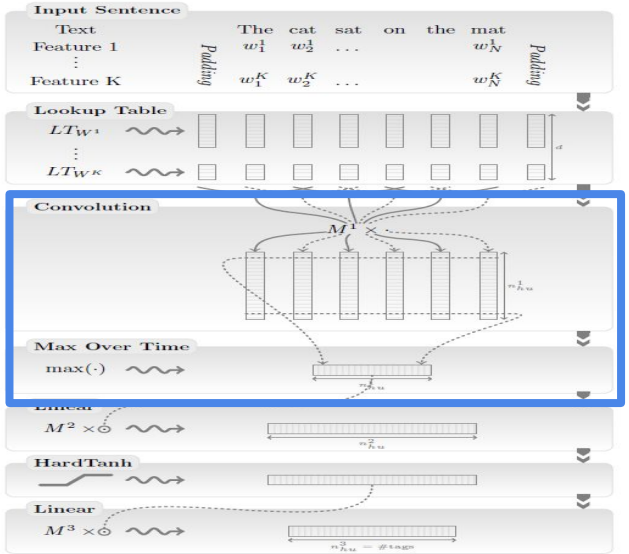
- traditional way:
  Syntactic parsing → pruning candidate arguments →
  argument recognition → argument labeling → post processing
- deep neural network: e.g., bidirectional LSTM RNN, cell w/ dependency tree...
- ...and more

3. Networks | overview

The main idea is to transform NLP task into an optimization problem



Window Approach



Sentence Approach

3. Networks | word⟶ word feature vector

Lookup table layer

word vector

$$LT_W(w) = \langle W \rangle_w^1$$

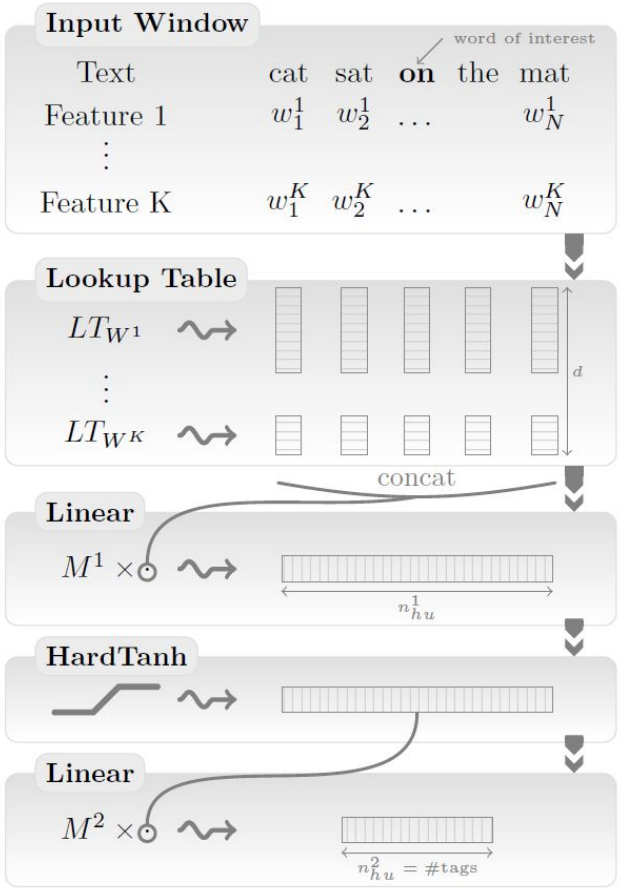
$$LT_W([w]_1^T) = \begin{pmatrix} \langle W \rangle_{[w]_1}^1 & \langle W \rangle_{[w]_2}^1 & \cdots & \langle W \rangle_{[w]_T}^1 \end{pmatrix}$$

word sequence

matrix

3. Networks | word feature vector → higher level feature

- Window Approach
    - Fixed Size Window
    - Look up Table
    - Linear Layer
    - Scoring



Lookup Table

$$f_\theta^1 = \langle LT_W([w]_1^T)\rangle_t^{d_{win}} = \begin{pmatrix} \langle W\rangle^1_{[w]_{t-d_{win}/2}} \\ \vdots \\ \langle W\rangle^1_{[w]_t} \\ \vdots \\ \langle W\rangle^1_{[w]_{t+d_{win}/2}} \end{pmatrix} \; d_{wrd}\, k_{sz}\text{-dimensional}$$

HardTanh

$$\left[f_\theta^l\right]_i = \text{HardTanh}\left(\left[f_\theta^{l-1}\right]_i\right)$$

$$\text{HardTanh}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 <= x <= 1 \\ 1 & \text{if } x > 1 \end{cases}$$

Scoring

3. Networks | word feature vector ⟶ higher level feature

Sentence approach

- Address the problem in SRL task-->takes the whole sentence instead of a window of words as input
- Most layers similar to window approach
- Additional layer: Convolution layer & Max layer

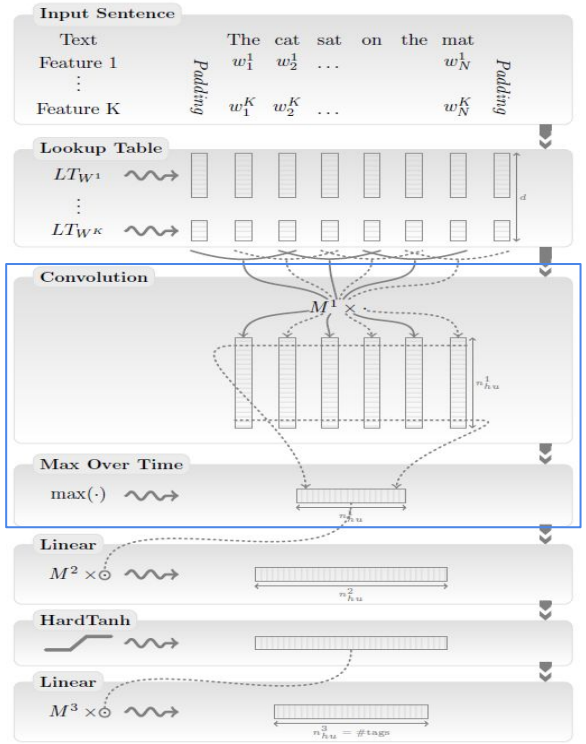
Matrix W same for all windows

Convolutional Layer:

$$\langle f_\theta^l \rangle_t^1 = W^l \langle f_\theta^{l-1} \rangle_t^{d_{win}} + b^l \quad \forall t$$

Max Layer:

$$\left[f_\theta^l \right]_i = \max_t \left[f_\theta^{l-1} \right]_{i,t} \qquad 1 \le i \le n_{hu}^{l-1}$$



3. Networks | training

- Max likelihood over training data $\theta \mapsto \sum_{(x,y)\in\mathcal{T}} \log p(y \mid x, \theta)$
- Stochastic gradient ascent
  - A random sample is drawn from dataset to make parameter update
- Two loss functions
  - Word-level log-likelihood
  - Sentence-level log-likelihood
    - Some tag pairs are unlikely to occur for grammatical reasons
    - Capture the tag dependency in some tasks

3. Networks | training

- Same as loss used in multiclass classification
- Word-level log-likelihood (WLL)
  - Non-negativity
  - Unity
- Use log-likelihood to ease the implementation of backpropagation

$$p(i|x,\theta) = \frac{e^{[f_\theta]_i}}{\sum_j e^{[f_\theta]_j}}$$

$$\log p(y|x,\theta) = [f_\theta]_y - \log(\sum_i e^{z_i})$$

$$\log p(y|x,\theta) = [f_\theta]_y - \log(\sum_j e^{[f_\theta]_j})$$

$$\log p(i|x,\theta) = [f_\theta]_i - \log(\sum_j e^{[f_\theta]_j})$$

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

3. Networks | training

- Sentence-level log-likelihood

score: (word sequence, tag sequence) pair

same as WLL

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

tag transition score for *successive* words *t*-1 & *t*

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

recursion + memoization,
$O(K^T) \rightarrow O(K^2 T)$

3. Networks | supervised benchmark results

Task	Window/Conv. size	Word dim.	Caps dim.	Hidden units	Learning rate
POS	$d_{win} = 5$	$d^0 = 50$	$d^1 = 5$	$n^1_{hu} = 300$	$\lambda = 0.01$
CHUNK	”	”	”	”	”
NER	”	”	”	”	”
SRL	”	”	”	$n^1_{hu} = 300$ $n^2_{hu} = 500$	”

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	55.40
NN+SLL	96.37	90.33	81.47	70.99

NN: neural network

WLL: word-level level log-likelihood

SLL: sentence-level log-likelihood

3. Networks | glimpse at word embedding space/lookup table

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

"Ideally, we would like semantically similar words to be close in the embedding space represented by the word lookup table: by continuity of the neural network function, tags produced on semantically similar sentences would be similar."

4. Techniques to improve the performance of NN

1) Unlabeled data
2) Multi-task learning
3) Task-specific engineering

4.1 Unlabeled data (word embeddings)

- Model performance hinges on initialization of look-up table
- Use unlabeled data to improve embedding
  - 221 million unlabeled words versus 130 thousand labeled words
  - 1700 times of the unlabeled data than labeled ones
- System
  - Use window approach architecture to acquire word embedding
  - Use ranking criterion instead of entropy (SLL and WLL previously described)
    - Computing normalization term is demanding even with efficient algorithm
    - Entropy criterion tends to ignore infrequent words
      - 15% of the most common words appear about 90% of the time

$$\theta \mapsto \sum_{x \in X} \sum_{w \in \mathcal{D}} \max \left\{ 0 , 1 - f_\theta(x) + f_\theta(x^{(w)}) \right\}$$

4.1 Unlabeled data | ranking criterion

- Same as multiclass hinge loss in SVM

$$\min_{\theta} \sum_{j \neq y_i} \max\{0, 1 + s_j(\theta) - s_{y_i}(\theta)\}$$

- Enforce the non-negativity of loss

$$\max\{0, 1 - (f_\theta(x) - f_\theta(x^{(w)}))\}$$

- Equivalent to maximize difference (margin) between
  - Score of sequence with true word
  - Score of sequence with replaced word

$$f_\theta(x) - f_\theta(x^{(w)})$$

$$\min_{\theta} \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{D}} \max\{0, 1 + f_\theta(x^{(w)}) - f_\theta(x)\}$$

$x^{(w)}$ is the sentence obtained by replacing central word of sequence $\{w_1, w_2, \cdots, w_{d_{win}}\}$ with a different word w.

- Semantic meaning of individual word is then distinguishable from others in this way.

4.1 Unlabeled data | embeddings

Appealing: *"syntactic and semantic properties of the neighbors are clearly related to those of the query word"*

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

4.1 Unlabeled data | *semi*-supervised benchmark results

initializing the word lookup tables of the supervised networks with the embeddings computed by the language models

Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40
NN+SLL	96.37	90.33	81.47	70.99
NN+WLL+LM1	97.05	91.91	85.68	58.18
NN+SLL+LM1	97.10	93.65	87.58	73.84
NN+WLL+LM2	97.14	92.04	86.96	58.34
NN+SLL+LM2	97.20	93.63	88.67	74.15

supervised training stage is free to *modify* the lookup tables

WLL: word-level level log-likelihood

SLL: sentence-level log-likelihood

NN: neural network

LM: language model

4.2 Multi-task learning

- features trained for one task can be useful for related tasks
  - e.g., language model → word embeddings → supervised networks for NLP

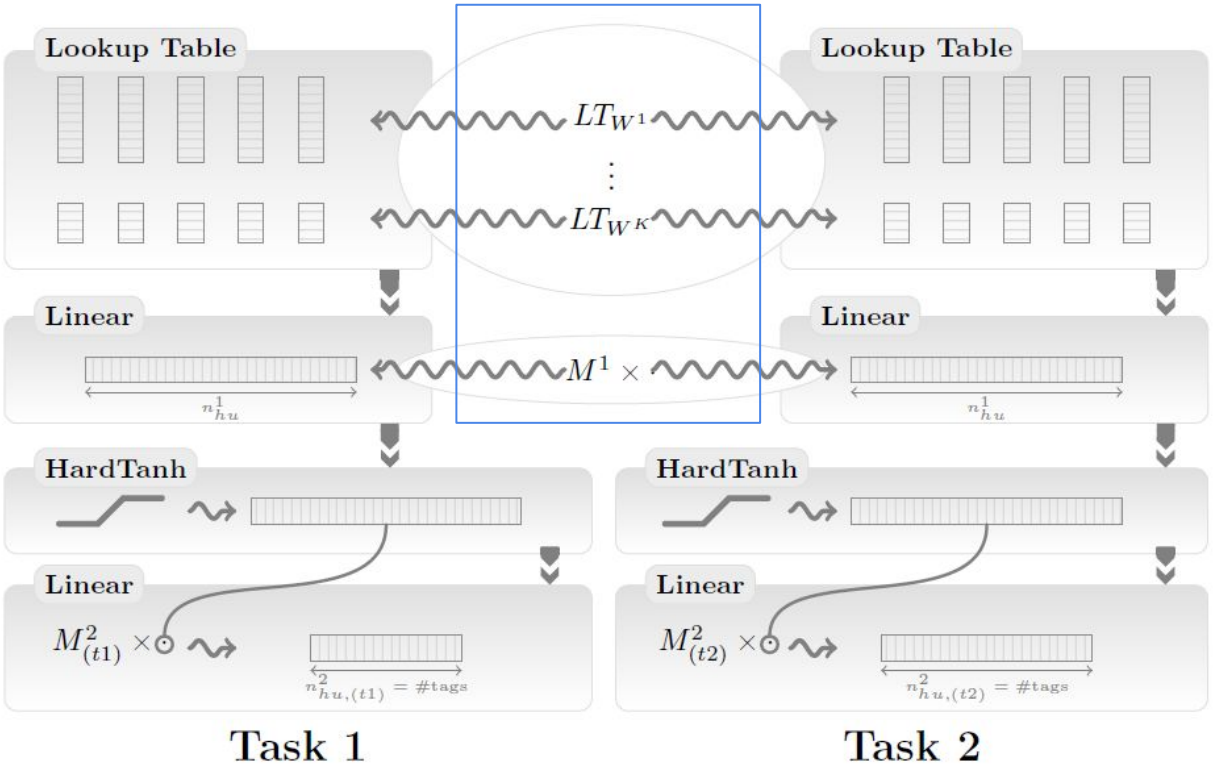
more systematic way:

- models of all tasks are jointly trained with an additional *linkage* between trainable parameters (to improve generalization error)
  - regularization in joint cost function biasing from common representations
  - shared certain parameters defined a priori

4.2 Multi-task learning | multitasking example

shared parametres:

- lookup table
- 1st linear layer
- (convolution layer)



training:

minimizing the loss *averaged* across all tasks

4.2 Multi-task learning | multi-task benchmark results

Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
	Window Approach			
NN+SLL+LM2	97.20	93.63	88.67	–
NN+SLL+LM2+MTL	97.22	94.10	88.62	–
	Sentence Approach			
NN+SLL+LM2	97.12	93.37	88.78	74.15
NN+SLL+LM2+MTL	97.22	93.75	88.27	74.29

- produce a single *unified* network that performs well for all these tasks (sentence approach)
- only lead to *marginal* improvements over using a separate network for each task

4.3 The temptation: Gazetteers

- Traditional NER task
- Restricted to the Gazetteers provided by the CONLL challenge
- Not only words but chunks
- Resulting Performance
- Explanation

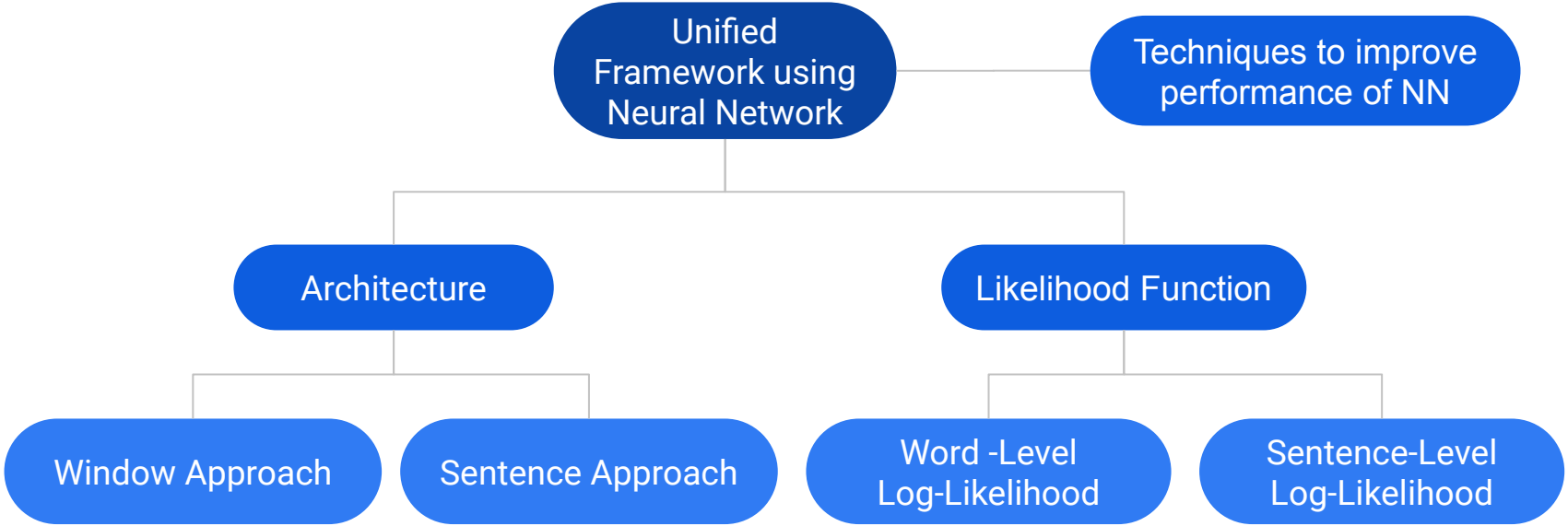
Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL
Benchmark Systems	97.24	94.29	89.31	77.92
NN+SLL+LM2	97.20	93.63	88.67	74.15
NN+SLL+LM2+Suffix2	97.29	–	–	–
NN+SLL+LM2+Gazetteer	–	–	89.59	–
NN+SLL+LM2+POS	–	94.32	88.67	–
NN+SLL+LM2+CHUNK	–	–	–	74.72

4.3 The temptation: Engineering a Sweet Spot

- Standalone Version of the Architecture in terms of 4 tasks
- “SENNA”
- Performance

Task		Benchmark	SENNA
Part of Speech (POS)	(Accuracy)	97.24 %	97.29 %
Chunking (CHUNK)	(F1)	94.29 %	94.32 %
Named Entity Recognition (NER)	(F1)	89.31 %	89.59 %
Parse Tree level 0 (PT0)	(F1)	91.94 %	92.25 %
Semantic Role Labeling (SRL)	(F1)	77.92 %	75.49 %

5. Conclusion



6. Historical remark

- This paper is regarded as a **milestone** in solving NLP problem using neural network approach.
  - Bengio et al. 2003 - A Neural Probabilistic Language Model (covered by professor)
  - Collobert et al. 2011 - Natural Language Processing (Almost) from Scratch (covered in this class)
  - Mikolov et al. 2013 - Distributed Representations of Words and Phrases and their Compositionality (aka Word2Vec) (covered by professor)

Q & A

Thank you 😄

The networks | word feature vector ⟶ higher level feature

Tagging schemes

- POS:
- NER, CHUNK, SRL:

Scheme	Begin	Inside	End	Single	Other
IOB	B-X	I-X	I-X	B-X	O
IOE	I-X	I-X	E-X	E-X	O
IOBES	B-X	I-X	E-X	S-X	O

4.1 Unlabeled data | data sets

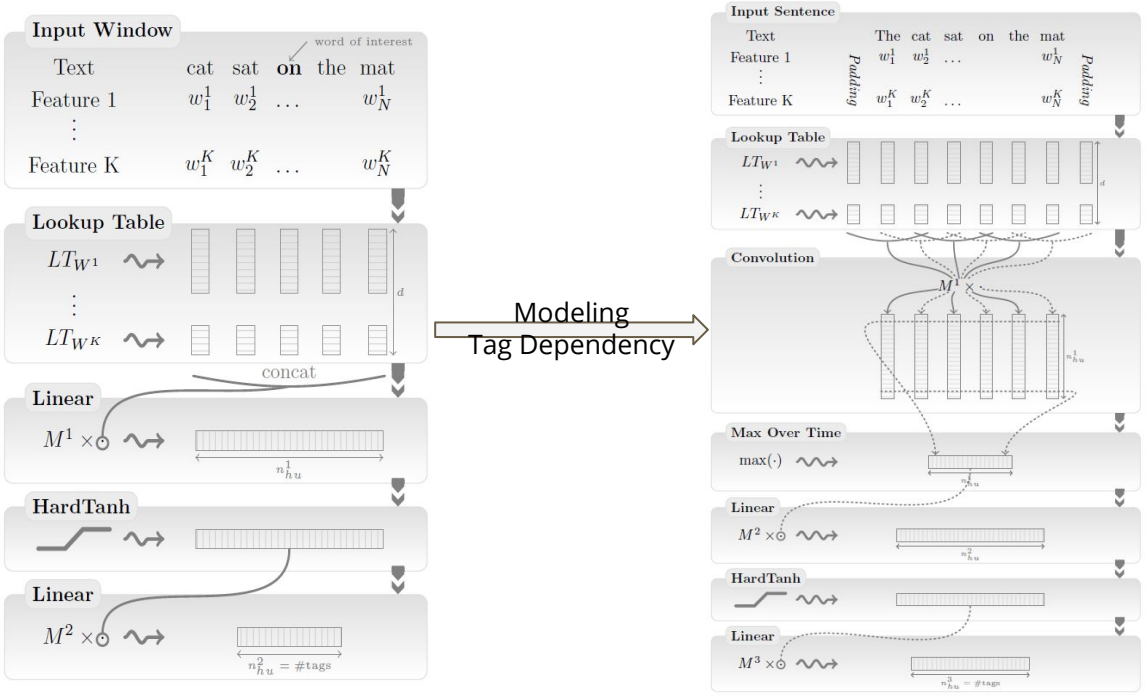
- Entire English Wikipedia
- Reuters RCV1

↓

- LM1: Wikipedia
- LM2: Wikipedia + Reuters

3. Networks | design philosophy

- First try simplest possible architecture to solve problems at hand.
- Then add additional component to address potential issues



Tagging *"The old man the boat"*.

2. Benchmark tasks | evaluation and discussion

Tasks	Benchmark systems	Evaluation
Part-of-speech tagging (POS)	Toutanova et al. (2003)	*per-word* accuracy
Chunking (CHUNK)	Sha and Pereira (2003)	F1 score
Named entity recognition (NER)	Ando and Zhang (2005)	F1 score
Semantic role labeling (SRL)	Koomen et al. (2005)	F1 score

4.3 The temptation (task-specific engineering)

Many subsections:

- Suffix features
- **Gazetteers**
- Cascading
- Ensembles
- **Parsing**
- Word representations
- Engineering a sweet spot

Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL
Benchmark Systems	97.24	94.29	89.31	77.92
NN+SLL+LM2	97.20	93.63	88.67	74.15
NN+SLL+LM2+Suffix2	97.29	–	–	–
NN+SLL+LM2+Gazetteer	–	–	89.59	–
NN+SLL+LM2+POS	–	94.32	88.67	–
NN+SLL+LM2+CHUNK	–	–	–	74.72

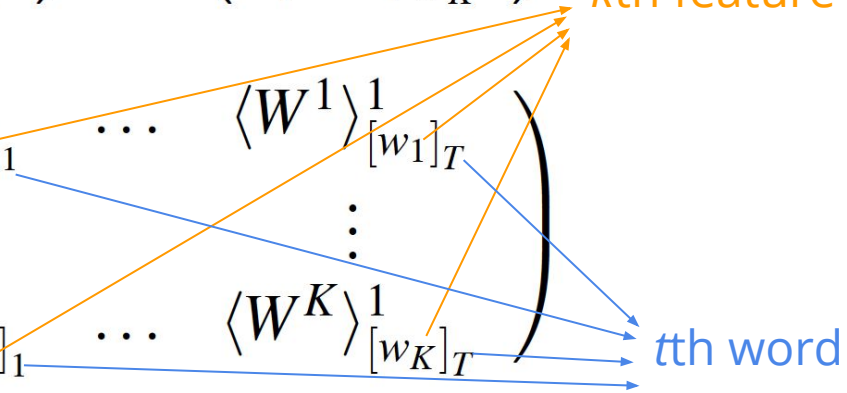
一个人讲

3. Networks | word⟶ word feature vector

Extending to any discrete features

$$LT_{W^1,\ldots,W^K}(w) = \begin{pmatrix} LT_{W^1}(w_1) \\ \vdots \\ LT_{W^K}(w_K) \end{pmatrix} = \begin{pmatrix} \langle W^1 \rangle^1_{w_1} \\ \vdots \\ \langle W^K \rangle^1_{w_K} \end{pmatrix}$$

$$LT_{W^1,\ldots,W^K}([w]_1^T) = \begin{pmatrix} \langle W^1 \rangle^1_{[w_1]_1} & \cdots & \langle W^1 \rangle^1_{[w_1]_T} \\ \vdots & & \vdots \\ \langle W^K \rangle^1_{[w_K]_1} & \cdots & \langle W^K \rangle^1_{[w_K]_T} \end{pmatrix}$$



3. Networks | notations

Layer l: function $f_\theta^l(\cdot)$

Matrix A: coefficient $[A]_{i,j}$, “concatenated” vector $\langle A \rangle_i^{d_{win}}$

Vector v: scalar $[v]_i$

Sequence of elements $[x]_1^T$: element $[x]_i$

$$\{x_1, x_2, \ldots, x_T\}$$

3. Networks | training

word *t*, tag *k*

$$\delta_t(k) \triangleq \underset{\{[j]_1^t \cap [j]_t = k\}}{\text{logadd}} \; s([x]_1^t, [j]_1^t, \tilde{\theta})$$

associativity

$$= \underset{i}{\text{logadd}} \; \underset{\{[j]_1^t \cap [j]_{t-1} = i \cap [j]_t = k\}}{\text{logadd}} \; s([x]_1^t, [j]_1^{t-1}, \tilde{\theta}) + [A]_{[j]_{t-1},k} + [f_\theta]_{k,t}$$

distributivity

$$= \underset{i}{\text{logadd}} \, \delta_{t-1}(i) + [A]_{i,k} + [f_\theta]_{k,t}$$

$$= [f_\theta]_{k,t} + \underset{i}{\text{logadd}} \left(\delta_{t-1}(i) + [A]_{i,k} \right) \quad \forall k,$$

word *t*-1, *all* tags

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

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

linear time $O(K^2T)$, where *K* is #tags?

tag \ word	1	...	*t*-1	*t*	...	*T*
1						
...						
k			?			
...						
K						

Critical discussion

- Goal
- Mean

Fast and efficient “all purpose” NLP tagger

- Avoid task-specific engineering as much as possible
- Rely on large unlabeled data sets
- Discover internal presentations