Natural Language Processing (almost) from Scratch

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, Pavel Kuksa (2011)

Presented by

Tara Vijaykumar
tgv2@illinois.edu

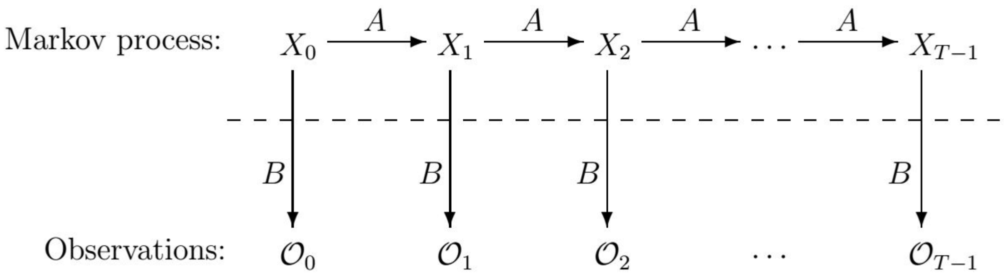
Content

Sequence Labeling

- assignment of a categorical label to each member of a sequence of observed values
- Eg: part of speech tagging

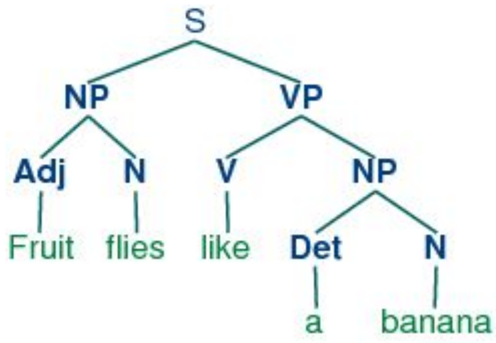
Mary	had	a	little	lamb
(noun)	(verb)	(det)	(adj)	(noun)

- can be treated as a set of independent classification tasks
  - choose the globally best set of labels for the entire sequence at once
- algorithms are probabilistic in nature
  - Markov assumption
  - Hidden Markov model (HMM)



POS Tagging

- Label word with syntactic tag (verb, noun, adverb…)
- best POS classifiers
  - trained on windows of text, which are then fed to bidirectional decoding algorithm during inference
  - Features - previous and next tag context, multiple words (bigrams, trigrams. . .) context
- **Shen et al. (2007)**
  - “Guided learning” - bidirectional sequence classification using perceptrons

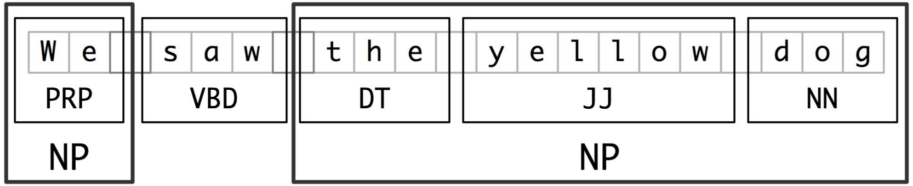



```
Agatha found that book interesting
  w1     w2    w3   w4      w5
```

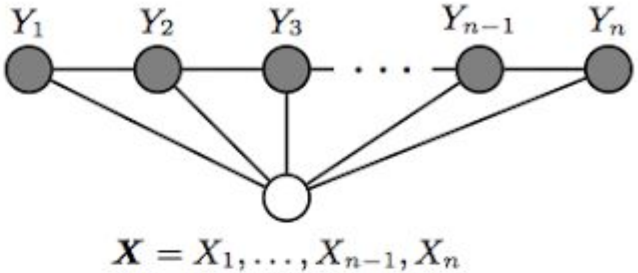
If we scan from left to right, we may find it difficult to resolve the ambiguity of the label for `that`, which could be either DT (determiner), or IN (preposition or subordinating conjunction) in the Penn Treebank. However, if we resolve the labels for `book` and `interesting`, it would be relatively easy to figure out the correct label for `that`.

Chunking

- labeling segments of a sentence with syntactic constituents (NP or VP)
- each word assigned only one unique tag, encoded as begin-chunk (B-NP) or inside-chunk tag (I-NP)
- evaluated using CoNLL shared task



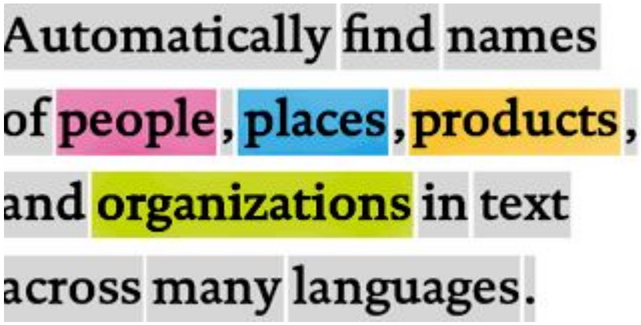
- **Sha and Pereira, 2003**
  - systems based on second-order random fields
  - Conditional Random Fields



Named Entity Recognition

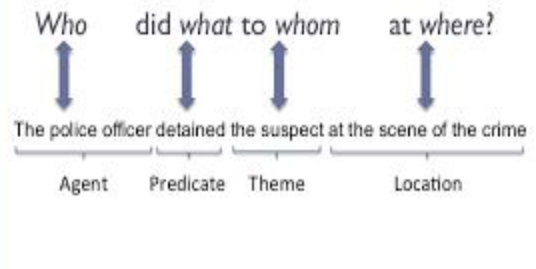
- labels atomic elements in the sentence into categories (“PERSON”, “LOCATION”)

- **Ando and Zhang (2005)**
  - semi-supervised approach
  - Viterbi decoding at test time
  - Features: words, POS tags, suffixes and prefixes or CHUNK tags

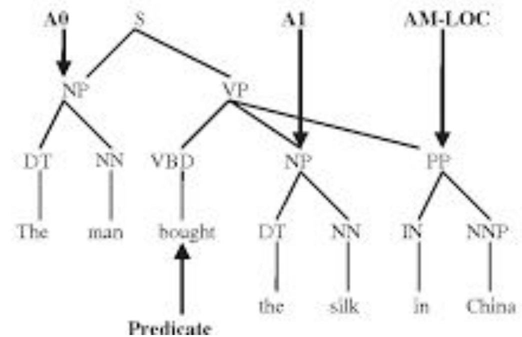


Semantic Role Labeling

- give a semantic role to a syntactic constituent of a sentence
- State-of-the-art SRL systems consist of stages
  - producing a parse tree
  - identifying which parse tree nodes represent the arguments of a given verb,
  - classifying nodes to compute the corresponding SRL tags



- **Koomen et al. (2005)**
  - takes the output of multiple classifiers and combines them into a coherent predicate-argument output
  - optimization stage takes into account recommendation of the classifiers and problem specific constraints



Introduction

- Existing systems
  - Find intermediate representations with task-specific features
    - Derived from output of existing systems (runtime dependencies)
  - Advantage: effective due to extensive use of linguistic knowledge
  - How to progress toward broader goals of NL understanding?

- Collobert et al. 2011
  - Single learning system to discover internal representations
  - Avoid large body of linguistic knowledge - instead, transfer intermediate representations discovered on large unlabeled data sets
  - “Almost from scratch” - reduced reliance on prior NLP knowledge

Remarks

- comparing systems
  - do not learn anything of the quality of each system if they were trained with different labeled data
  - refer to benchmark systems - top existing systems which avoid usage of external data and have been well-established in the NLP field

- for more complex tasks (with corresponding lower accuracies), best systems have more engineered features
  - POS task is one of the simplest of our four tasks, and only has relatively few engineered features
  - SRL is the most complex, and many kinds of features have been designed for it

Networks

- Traditional NLP approach
  - extract rich set of hand-designed features (based on linguistic intuition, trial and error)
    - task dependent
  - Complex tasks (SRL) then require a large number of possibly complex features (eg: extracted from a parse tree)
    - can impact the computational cost

- Proposed approach
  - pre-process features as little as possible - make it generalizable
  - use a multilayer neural network (NN) architecture trained in an end-to-end fashion.

Transforming Words into Feature Vectors

- For efficiency, words are fed to our architecture as indices taken from a finite dictionary D.
- The first layer of our network maps each of these word indices into a feature vector, by a lookup table operation. Initialize the word lookup table with these representations (instead of randomly)

- For each word w ∈ D , an internal d_{wrd} -dimensional feature vector representation is given by the lookup table layer LTW (·):
    - $LT_W(w) = \langle W \rangle_w^1 ,$

      where W is a matrix of parameters to be learned, ⟨W⟩ is the w^{th} column of W and d_{wrd} is the word vector size (a hyper-parameter)
- Given a sentence or any sequence of T words, the output matrix produced -

$$LT_W([w]_1^T) = \left(\langle W \rangle_{[w]_1}^1 \quad \langle W \rangle_{[w]_2}^1 \quad \dots \quad \langle W \rangle_{[w]_T}^1 \right)$$

Extracting Higher Level Features from Word Feature Vectors

- **Window approach:** assumes the tag of a word depends mainly on its neighboring words
- Word feature window given by the first network layer:

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

- Linear Layer:

$$f_\theta^l = W^l f_\theta^{l-1} + b^l$$

- HardTanh Layer:

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

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

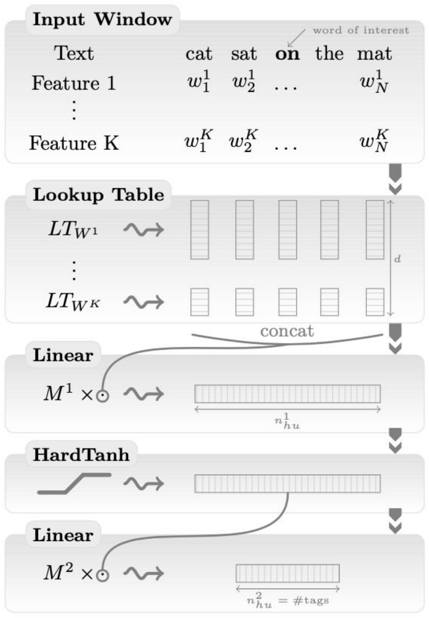


Figure 1: Window approach network.

- Scoring: size of number of tags with corresponding score
- Feature window is not well defined for words near the beginning or the end of a sentence - augment the sentence with a special "PADDING" akin to the use of "start" and "stop" symbols in sequence models.

Extracting Higher Level Features from Word Feature Vectors

- **Sentence approach:** window approach fails with SRL, where the tag of a word depends on a verb chosen beforehand in the sentence
- Convolutional Layer: generalization of a window approach - for all windows t, output column of l^{th} 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:
  - average operation does not make much sense - most words in the sentence do not have any influence on the semantic role of a given word to tag.
  - max approach forces the network to capture the most useful local features

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

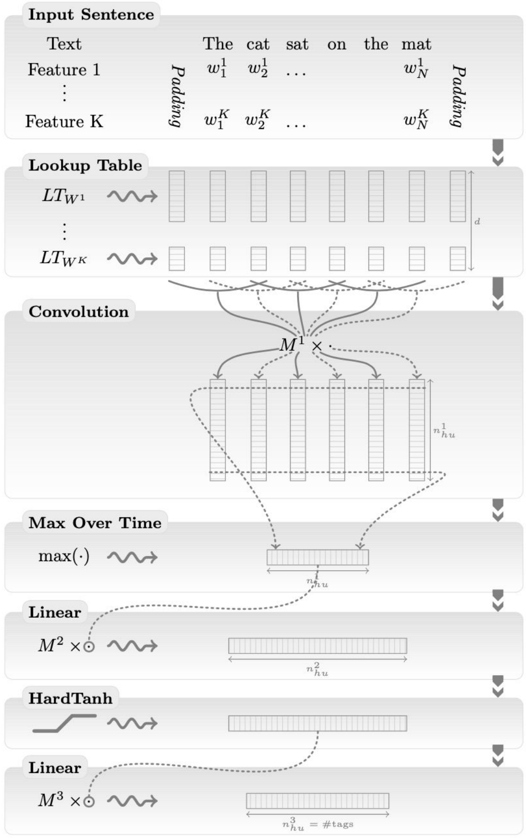


Figure 2: Sentence approach network.

Extracting Higher Level Features from Word Feature Vectors

- **Tagging schemes:**
  - window approach
    - tags apply to the word located in the center of the window
  - sentence approach
    - tags apply to the word designated by additional markers in the network input

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

Table 3: Various tagging schemes. Each word in a segment labeled "X" is tagged with a prefixed label, depending of the word position in the segment (begin, inside, end). Single word segment labeling is also output. Words not in a labeled segment are labeled "O". Variants of the IOB (and IOE) scheme exist, where the prefix B (or E) is replaced by I for all segments not contiguous with another segment having the same label "X".

- most expressive IOBES tagging scheme

Training

- For θ trainable parameters and a training set T: maximize the following log-likelihood with respect to θ:

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

- **Stochastic gradient:** maximization is achieved by iteratively selecting a random example (x, y) and making a gradient step:

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

- **Word-level log likelihood**: each word in sentence is considered <u>independently</u>
  Get conditional tag probability with use of softmax

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

Training

- Introduce scores:
  - Transition score $[A]_{ij}$: from i to j tags in successive words
  - Initial score $[A]_{i0}$: starting from the ith tag

- **Sentence-level log likelihood**: enforces dependencies between the predicted tags in a sentence.
  - Score of sentence along a path of tags, using initial and transition scores

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

  - Maximize this score
    - Viterbi algorithm for inference

$$\underset{[j]_1^T}{\operatorname{argmax}}\, s([x]_1^T, [j]_1^T, \tilde{\theta})$$

Results

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

Table 4: Comparison in generalization performance of benchmark NLP systems with a vanilla neural network (NN) approach, on POS, chunking, NER and SRL tasks. We report results with both the word-level log-likelihood (WLL) and the sentence-level log-likelihood (SLL). Generalization performance is reported in per-word accuracy rate (PWA) for POS and F1 score for other tasks. The NN results are behind the benchmark results, in Section 4 we show how to improve these models using unlabeled data.

- Remarks:
  - **Architecture**: choice of hyperparameters such as the number of hidden units has a limited impact on the generalization performance
  - Prefer semantically similar words to be close in the embedding space represented by the word lookup table but that it is not the case

NLP (Almost) From Scratch Pt. 2

Harrison Ding

Word Embeddings

- Goal
  - Obtain Word Embeddings that can capture syntactic and semantic differences

FRANCE 454	JESUS 1973	XBOX 6909	REDDISH 11724	SCRATCHED 29869	MEGABITS 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

Datasets

- English Wikipedia (631 million words)
    - Constructed a dictionary of 100k most common words in WSJ
    - Replace the non-dictionary words with “RARE” tokens

- Reuters RCV1 Dataset (221 million words)
    - Extended dictionary to a size of 130k words where 30k were Reuters most common words

Ranking Criterion

- Cohen et al. 1998
  - Binary Preference Function
  - Ranking ordering
- Training is done with a windowed approach

X = Set of all possible text windows

D = All words in the dictionary

$x^{(w)}$ = Text window with the center word replaced by the chosen word

f(x) = Score of the text window

$$\forall x \in X \ \forall w \in D \max(0, 1 - f(x) + f(x^{(w)}))$$

Result of Embeddings for LM1

- Goal of capturing semantic and syntactic differences appears to have been achieved

FRANCE 454	JESUS 1973	XBOX 6909	REDDISH 11724	SCRATCHED 29869	MEGABITS 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

Tricks with Training

- Length of time calculated in weeks
- Problem
  - Difficult to try a large number of hyperparameter combinations
- Efficient Solution
  - Train networks based on earlier networks
  - Construct embeddings based on small dictionaries and use the best from there
  - “Breeding”

Language Models Information

- Language Model LM1
  - Window size $d_{win} = 11$
  - Hidden layer $n_{hu}^{1} = 100$ units
  - English Wikipedia
  - Dictionary sizes: 5k, 10k, 30k, 50k, 100k
  - Training time: 4 weeks

Language Models Information

- Language Model LM1
  - Window size $d_{win} = 11$
  - Hidden layer $n_{hu}^{1} = 100$ units
  - English Wikipedia
  - Dictionary sizes: 5k, 10k, 30k, 50k, 100k
  - Training time: 4 weeks
- Language Model LM2
  - Same dimensions as LM1
  - Initialized embeddings LM1
  - English Wikipedia + Reuters
  - Dictionary size: 130k
  - Training time: 3 more weeks

Comparison of Generalization Performance

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

Comparison of Generalization Performance

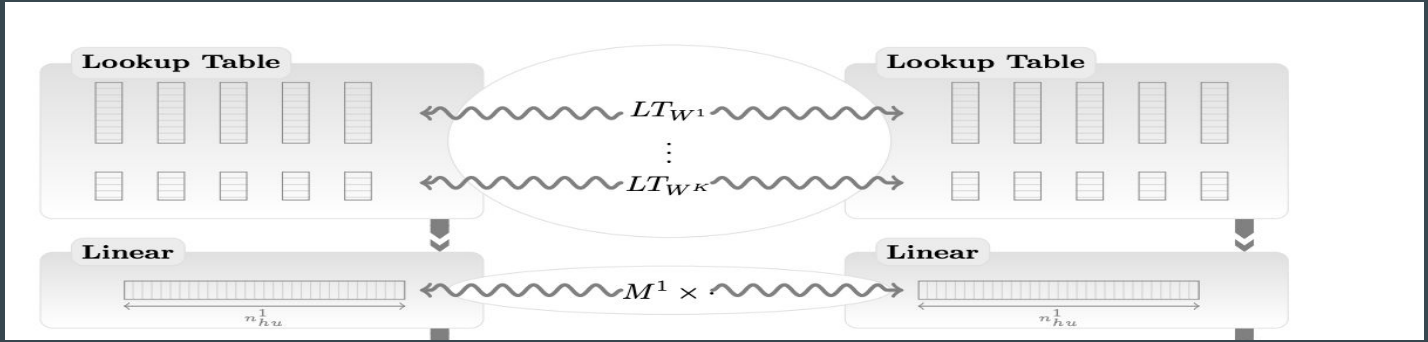
Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40
NN+SLL	96.37	90.33	81.47	70.99
NN+WLL+LM1	97.05	91.91	85.68	58.18
NN+SLL+LM1	97.10	93.65	87.58	73.84

Comparison of Generalization Performance

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

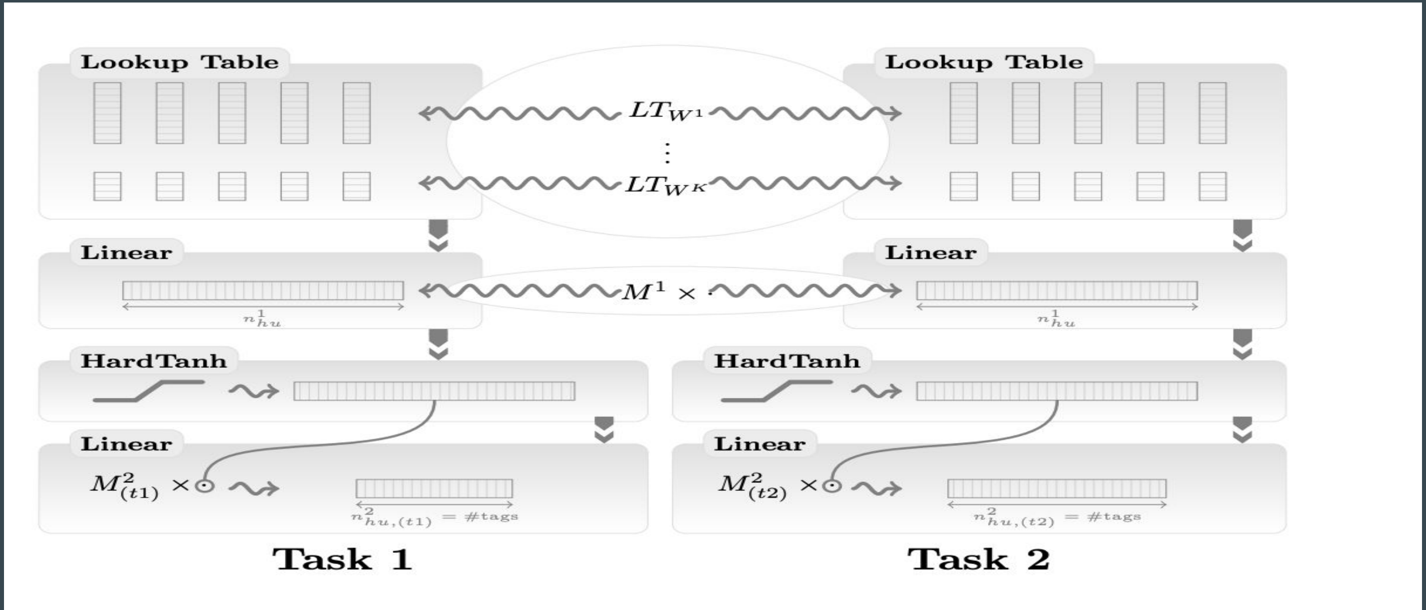
Multi-Task Learning

- Joint training = Training a neural network for two tasks
    - Easy to do when similar patterns appear in training tasks with different labels



Multi-Task Learning

- Joint training = Training a neural network for multiple tasks
  - Easy to do when similar patterns appear in training tasks with different labels



Results of Multi-Task Learning

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	–

Results of Multi-Task Learning

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

Adding a Task-Specific Features

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

Some other testing stuff later…

…

With parse trees and Brown Clusters…

Final Results and Putting It All Together

- Semantic/syntactic Extraction using a Neural Network Architecture (SENNA)

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 %

Concluding Information

- The NN technology is simple
  - Existed over twenty years before this paper was written
  - Simply used a neural network to do most of the work
- Conclusion
  - Throwing a bunch of unlabeled data at a neural network that is constructed correctly will yield state-of-the-art results (10 years ago)
- Fun fact
  - If they tried implementing this paper ten years prior to when it was written, it would probably finish in 10 years

Questions?

Citations

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. JMLR, 12:2493–2537.

END-TO-END SEQUENCE LABELING VIA BI-DIRECTIONAL LSTM-CNNS-CRF

Xuezhe Ma and Eduard Hovy
Presenter: Jiaxin Huang
03/13/2020

Advantages of Neural Sequence Models

- Prior Approaches
  - *Hand-crafted features: word spelling, orthographic features*
  - *Task-specific resources: external dictionaries*
  - *Linear statistical models: HMM, CRF*

Advantages of Neural Sequence Models

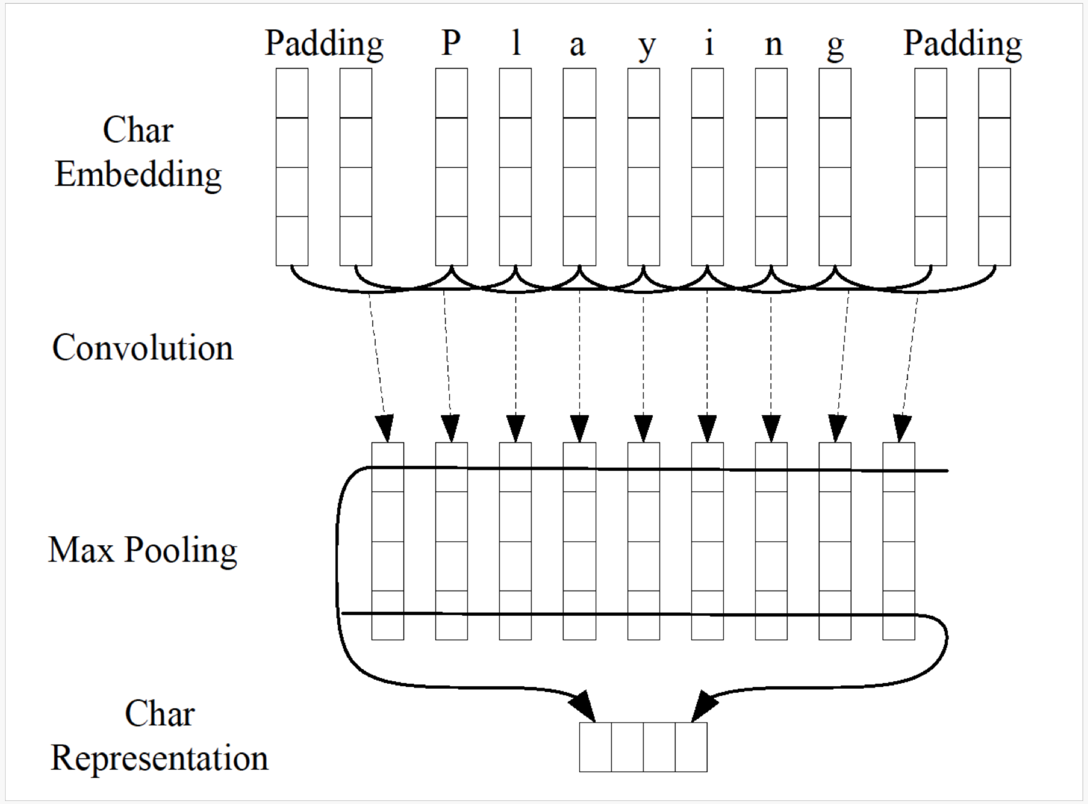
- Neural Sequence Models (in this paper)
  - *No hand-engineered features*
  - *No specialized knowledge resources*
  - *No data preprocessing beyond unsupervised word embedding training*

Neural Network Architecture

- Data Preparation
  - *NER Tag Schema used: BIOES instead of BIO*
    - B: Beginning
    - I: Inside
    - E: End
    - O: Outside
    - S: Single
  - *Pre-trained Word Embeddings: Mapping from words to low-dimensional vectors*
    - GloVe
    - Word2Vec
    - Senna

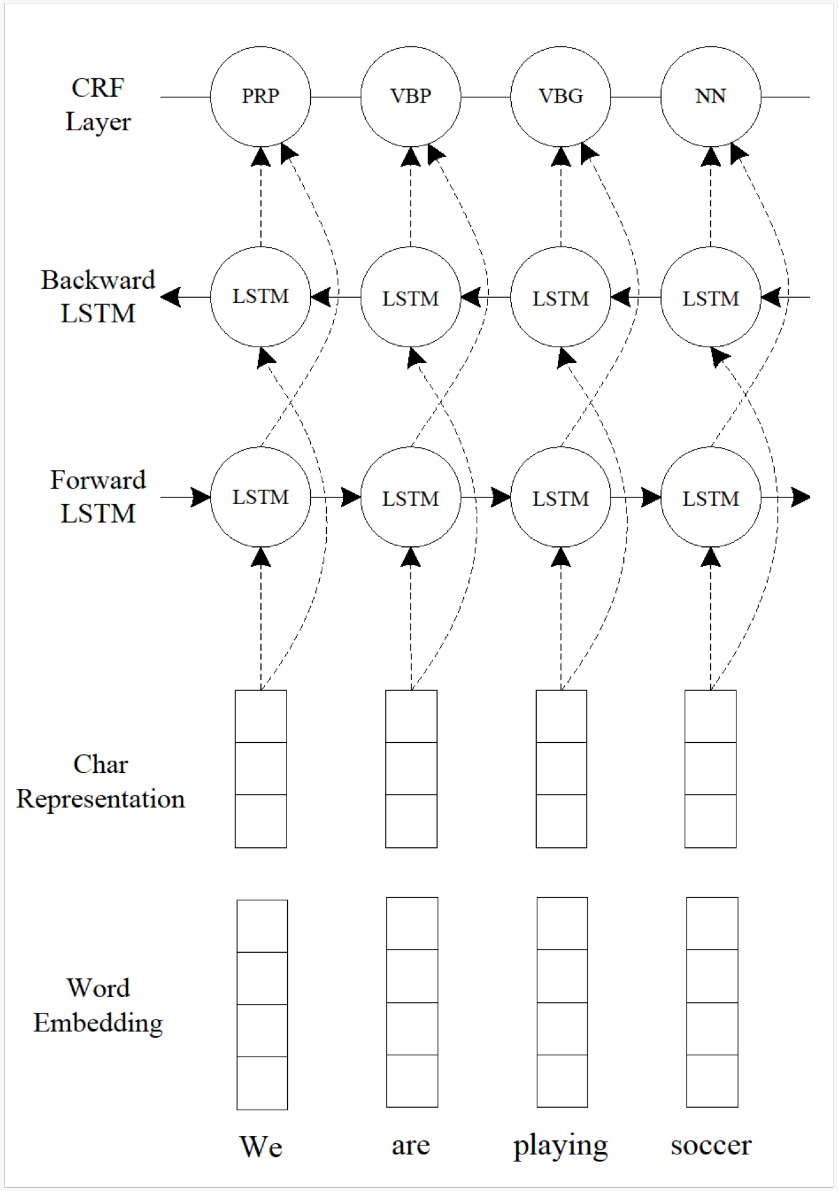
Neural Network Architecture

- CNN Encoder for Character-Level Representation
  - *A convolution layer on top of char embeddings to extract morphological information (like prefix or suffix of a word)*
  - *A dropout layer is applied before CNN.*



Neural Network Architecture

- Bi-directional LSTM for word-level encoding
  - *The word embedding and character-level representation are concatenated together as word-level representation.*
  - *The forward LSTM reads the sequence from left to right and generates a vector representing what it has seen so far.*
  - *The backward LSTM does the same in an opposite direction.*
- CRF layer (next page)
  - *Since the decisions of tags are not independent and can heavily depend on neighbors, we use a conditional random field to jointly label the sequence.*



Graphical Models

One hidden variable
E.g., document classification

A sequence of hidden Variables
E.g., NER, POS tagging

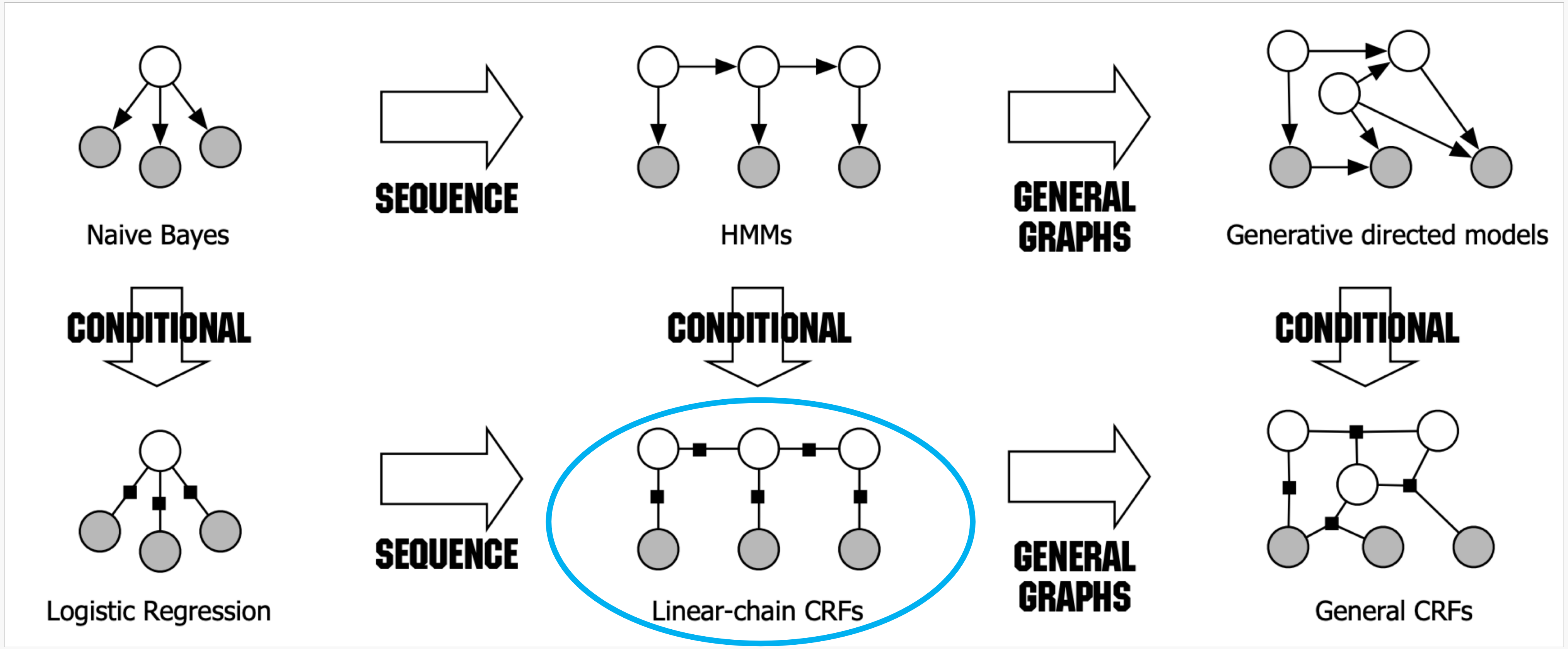
More general cases

Generative Models:

P(x, y)

Discriminative Models:

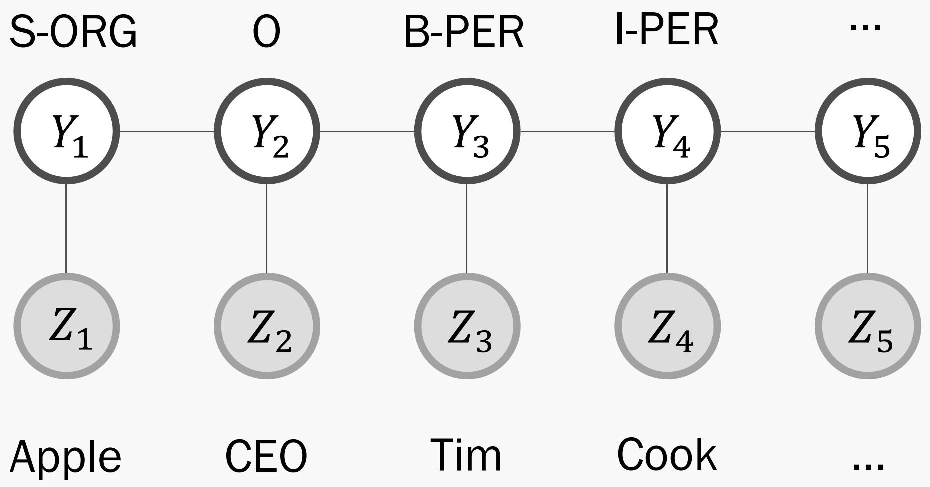
P(y|x)



Relationship between different graphical models.
Transparent nodes are hidden variables (labels), and grey nodes are observed words.

Linear-Chain CRF

- Linear-Chain CRF (Conditional Random Field) maximizes the conditional probability of a sequence of tags given the input sentence.



Y_i is the hidden variable (tag of words).

Z_i is the observation (word in the sentence).

- Softmax over all possible sequences of labels, with **y** being the tag sequence, and **z** being the input sentence.

$$p(\boldsymbol{y}|\mathbf{z};\mathbf{W},\mathbf{b}) = \frac{\prod_{i=1}^{n} \psi_i(y_{i-1}, y_i, \mathbf{z})}{\sum_{y' \in \mathcal{Y}(\mathbf{z})} \prod_{i=1}^{n} \psi_i(y'_{i-1}, y'_i, \mathbf{z})}$$

Numerator: score of a tag sequence factored into potential functions of subgraphs.

Denominator: sum over scores of all tag sequences.

Linear-Chain CRF in Neural Networks

- How potential functions are represented in neural networks:

$$\psi_i(y', y, \mathbf{z}) = \exp(\mathbf{W}_{y',y}^T \mathbf{z}_i + \mathbf{b}_{y',y})$$

  – $W_{y',y}^T$ *and* $\mathrm{b}_{y',y}$ *are the weight vector and bias corresponding to label pair* (y', y) *respectively.*

- CRF layer: Jointly decoding the best chain of labels of a given sequence.

$$\boldsymbol{y}^* = \underset{y \in \mathcal{Y}(\mathbf{z})}{\operatorname{argmax}}\, p(\boldsymbol{y}|\mathbf{z}; \mathbf{W}, \mathbf{b})$$

- Solving a sequence CRF model

  – *Training and decoding can be solved efficiently by adopting the Viterbi algorithm.*

Experiments —— Datasets

- POS tagging
  - *Wall Street Journal (Marcus et al., 1993)*
  - *Containing 45 different POS tags.*

- NER
  - *English data from CoNLL 2003 shared task (Tjong Kim Sang and De Meulder, 2003).*
  - *Four different types of named entities: PERSON, LOCATION, ORGANIZATION, and MISC.*

Experiments —— Ablation Study

	POS		NER					
	Dev	Test	Dev			Test		
Model	Acc.	Acc.	Prec.	Recall	F1	Prec.	Recall	F1
BRNN	96.56	96.76	92.04	89.13	90.56	87.05	83.88	85.44
BLSTM	96.88	96.93	92.31	90.85	91.57	87.77	86.23	87.00
BLSTM-CNN	97.34	97.33	92.52	93.64	93.07	88.53	90.21	89.36
BRNN-CNN-CRF	97.46	97.55	94.85	94.63	94.74	91.35	91.06	91.21

- BLSTM > BRNN
- CNN brings significant improvement: character level information is important for sequence labeling problems.
- CRF brings significant improvement: jointly decoding label sequences can significantly benefit the final performance.

Experiments —— Comparison w. Baselines

POS tagging accuracy.

Feed-forward
CharWNN

Model	Acc.
Giménez and Màrquez (2004)	97.16
Toutanova et al. (2003)	97.27
Manning (2011)	97.28
Collobert et al. (2011)‡	97.29
Santos and Zadrozny (2014)‡	97.32
Shen et al. (2007)	97.33
Sun (2014)	97.36
Søgaard (2011)	97.50
This paper	**97.55**

NER F1 score.

Model	F1	
Chieu and Ng (2002)	88.31	
Florian et al. (2003)	88.76	
Ando and Zhang (2005)	89.31	
Collobert et al. (2011)‡	89.59	
Huang et al. (2015)‡	90.10	BLSTM + CRF + features
Chiu and Nichols (2015)‡	90.77	BLSTM + CNN + features
Ratinov and Roth (2009)	90.80	
Lin and Wu (2009)	90.90	
Passos et al. (2014)	90.90	
Lample et al. (2016)‡	90.94	BLSTM for w & c + CRF
Luo et al. (2015)	91.20	
This paper	**91.21**	

- ‡ marks the neural models.

Experiments —— Other Model Designs

Results with different choices of word embeddings .

Embedding	Dimension	POS	NER
Random	100	97.13	80.76
Senna	50	97.44	90.28
Word2Vec	300	97.40	84.91
GloVe	100	**97.55**	**91.21**

Results with and w/o dropout.

	POS			**NER**		
	Train	**Dev**	**Test**	**Train**	**Dev**	**Test**
No	98.46	97.06	97.11	99.97	93.51	89.25
Yes	97.86	97.46	97.55	99.63	94.74	91.21

- NER relies more heavily on the quality of embeddings than POS tagging.
- GloVe > Senna > Word2Vec (vocabulary mismatch) > Random
- Dropout layers effectively reduce overfitting.

Experiments —— OOV Error Analysis

	POS							
	Dev				Test			
	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV
LSTM-CNN	97.57	**93.75**	90.29	80.27	97.55	**93.45**	90.14	80.07
LSTM-CNN-CRF	**97.68**	93.65	**91.05**	**82.71**	**97.77**	93.16	**90.65**	**82.49**
	NER							
	Dev				Test			
	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV
LSTM-CNN	94.83	87.28	96.55	82.90	90.07	89.45	100.00	78.44
LSTM-CNN-CRF	**96.49**	**88.63**	**97.67**	**86.91**	**92.14**	**90.73**	100.00	**80.60**

- Partition of words: in-vocabulary words (IV), out-of-training-vocabulary words (OOTV), out-of-embedding-vocabulary words (OOEV) and out-of-both-vocabulary words (OOBV)
- CRF layer for joint decoding helps improve the performance on words that are out of both the training and embedding sets. (OOBV)

Conclusion

- Advantages in Model Design of LSTM-CNNs-CRF:
  - *End-to-end model requiring no feature engineering and task-specific resources*
  - *Combining different levels of information by CNN and BLSTM*
  - *CRF layer is used to jointly decode the sequence.*
- Further Improvements:
  - *As embeddings are shown to greatly affect the performance of sequence labeling problems, efforts can be made to improve the quality of embeddings by multi-task learning.*
  - *For example, character level embedding is initialized randomly in this paper, but they can be improved by char-level language modeling, without further annotations.*

Neural Architectures for Named Entity Recognition

Author: Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, Chris Dyer

Presenter: Haoyang Wen

Named Entity Recognition

contentSkip to site indexPoliticsSubscribeLog InSubscribeLog InToday's PaperAdvertisementSupported ORG byF.B.I. Agent Peter Strzok PERSON , Who Criticized Trump PERSON in Texts, Is FiredImagePeter Strzok, a top F.B.I. GPE counterintelligence agent who was taken off the special counsel investigation after his disparaging texts about President Trump PERSON were uncovered, was fired. CreditT.J. Kirkpatrick PERSON for The New York TimesBy Adam Goldman ORG and Michael S. SchmidtAug PERSON . 13 CARDINAL , 2018WASHINGTON CARDINAL — Peter Strzok PERSON , the F.B.I. GPE senior counterintelligence agent who disparaged President Trump PERSON in inflammatory text messages and helped oversee the Hillary Clinton PERSON email and Russia GPE investigations, has been fired for violating bureau policies, Mr. Strzok PERSON 's lawyer said Monday DATE .Mr. Trump and his allies seized on the texts — exchanged during the 2016 DATE campaign with a former F.B.I. GPE lawyer, Lisa Page — in PERSON assailing the Russia GPE investigation as an illegitimate "witch hunt." Mr. Strzok PERSON , who rose over 20 years DATE at the F.B.I. GPE to become one of its most experienced counterintelligence agents, was a key figure in the early months DATE of the inquiry.Along with writing the texts, Mr. Strzok PERSON was accused of sending a highly sensitive search warrant to his personal email account.The F.B.I. GPE had been under immense political pressure by Mr. Trump PERSON to dismiss Mr. Strzok PERSON , who was removed last summer DATE from the staff of the special counsel, Robert S. Mueller III PERSON . The president has repeatedly denounced Mr. Strzok PERSON in posts on

Named Entity Recognition

- Challenges
  - Very small amount of data available for most languages and domains
  - Difficult to generalize from small sample of data

Named Entity Recognition

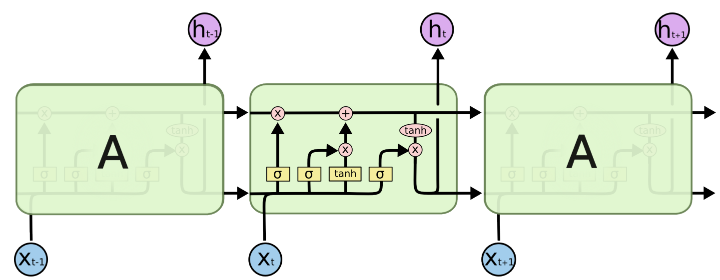
- Challenges
  - Very small amount of data available for most languages and domains
  - Difficult to generalize from small sample of data
- Results
  - Using constructed orthographic features
  - Using language-specific knowledge resources

Named Entity Recognition

- Challenges
  - Very small amount of data available for most languages and domains
  - Difficult to generalize from small sample of data
- Results
  - Using constructed orthographic features
  - Using language-specific knowledge resources
- This paper
  - Neural architectures for NER that
    - Uses **no** language-specific resources or features

Model I: LSTM-CRF

- LSTM
  - Input: A sequence of vectors
  - Return: another sequence that encoded every input vector with its context
- BiLSTM: for a given sentence $(x_1, x_2, \ldots, x_n)$
  - Compute $\overrightarrow{h_t}$ of the left context at every word t
  - Compute $\overleftarrow{h_t}$ of the right context at every word t
  - $h_t = [\, \overrightarrow{h_t}; \overleftarrow{h_t} \,]$
- BiLSTM as a sequence encoder



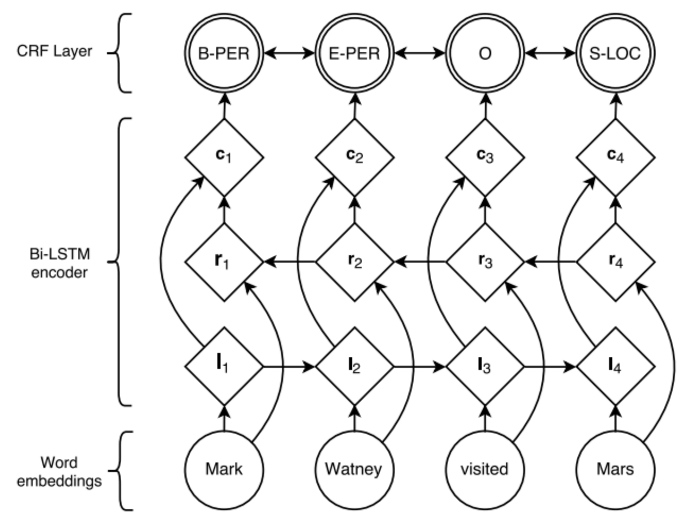
Model I: LSTM-CRF

- Naïve Tagging
  - Simply use h_t for each output y_t
    - independent tagging decision
    - Fail to capture strong dependencies between labels

- Modeling label dependency?

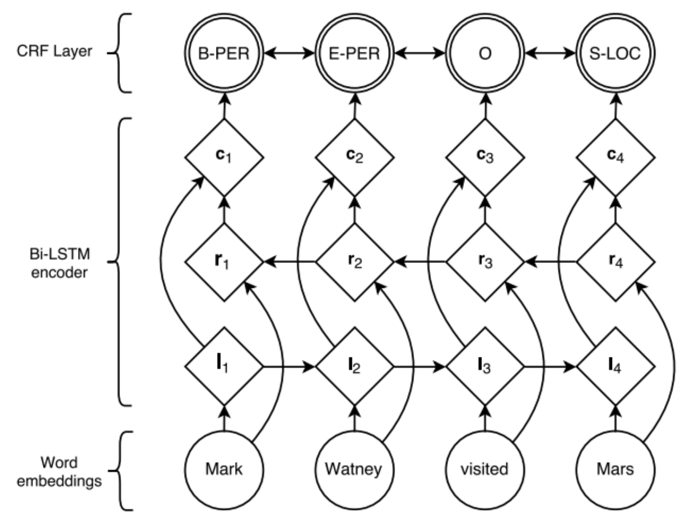
Model I: LSTM-CRF

- Naïve Tagging
- Conditional Random Field (CRF)
  - Consider $P \in \mathbb{R}^{n \times k}$ to be the matrix of scores output by BiLSTM
  - P_{ij}: the score of j^{th} tag of the i^{th} word



Model I: LSTM-CRF

- Naïve Tagging
- Conditional Random Field (CRF)
  - Consider $P \in \mathbb{R}^{n \times k}$ to be the matrix of scores output by BiLSTM
  - P_{ij}: the score of j^{th} tag of the i^{th} word
  - For a sequence of predictions $\boldsymbol{y} = (y_1, \ldots, y_n)$
    - Score over a sequence
      - $s(\boldsymbol{X}, \boldsymbol{y}) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i}$
    - $A_{y_i, y_{i+1}}$ is a score of transition from y_i to y_{i+1}
    - A softmax over all possible tag sequences



Model I: LSTM-CRF

- CRF Training
  - Maximize the log-probability

$$\log p(\boldsymbol{y}|\boldsymbol{X}) = \log \frac{e^{s(\boldsymbol{X},\boldsymbol{y})}}{\sum_{\widetilde{\boldsymbol{y}}\in \boldsymbol{Y_X}} e^{s(\boldsymbol{X},\widetilde{\boldsymbol{y}})}} = s(\boldsymbol{X},\boldsymbol{y}) - \log \sum_{\widetilde{\boldsymbol{y}}\in \boldsymbol{Y_X}} e^{s(\boldsymbol{X},\widetilde{\boldsymbol{y}})}$$

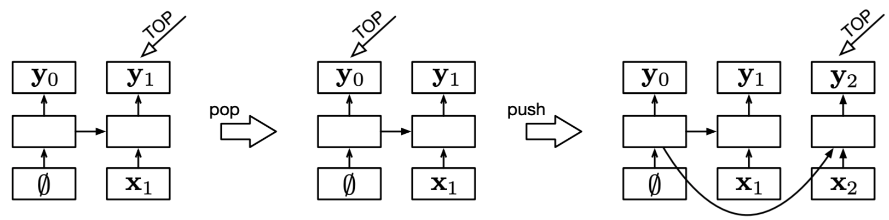
  - Dynamic Programming
- CRF Decoding

$$\boldsymbol{y}^* = \arg\max s(\boldsymbol{X},\boldsymbol{y})$$

  - Dynamic Programming

Model II: Chunking Algorithm

- Stack-LSTM (Dyer et al., 2015)



- Chunking Algorithm

$\mathbf{Out}_t$	$\mathbf{Stack}_t$	$\mathbf{Buffer}_t$	**Action**	$\mathbf{Out}_{t+1}$	$\mathbf{Stack}_{t+1}$	$\mathbf{Buffer}_{t+1}$	**Segments**
O	S	$(\mathbf{u}, u), B$	SHIFT	O	$(\mathbf{u}, u), S$	B	—
O	$(\mathbf{u}, u), \ldots, (\mathbf{v}, v), S$	B	REDUCE(y)	$g(\mathbf{u}, \ldots, \mathbf{v}, \mathbf{r}_y), O$	S	B	$(u \ldots v, y)$
O	S	$(\mathbf{u}, u), B$	OUT	$g(\mathbf{u}, \mathbf{r}_\varnothing), O$	S	B	—

Model II: Chunking Algorithm

- Transition sequence example

Transition	Output	Stack	Buffer	Segment
	[]	[]	[Mark, Watney, visited, Mars]	

Model II: Chunking Algorithm

- Transition sequence example

Transition	Output	Stack	Buffer	Segment
SHIFT	[]	[Mark]	[Watney, visited, Mars]	

Model II: Chunking Algorithm

- Transition sequence example

Transition	Output	Stack	Buffer	Segment
SHIFT	[]	[Mark, Watney]	[visited, Mars]	

Model II: Chunking Algorithm

- Transition sequence example

Transition	Output	Stack	Buffer	Segment
REDUCE(PER)	[(Mark Watney)-PER]	[]	[visited, Mars]	(Mark Watney)-PER

Model II: Chunking Algorithm

- Transition sequence example

Transition	Output	Stack	Buffer	Segment
OUT	[(Mark Watney)-PER, visited]	[]	[Mars]	

Model II: Chunking Algorithm

- Transition sequence example

Transition	Output	Stack	Buffer	Segment
SHIFT	[(Mark Watney)-PER, visited]	[Mars]	[]	

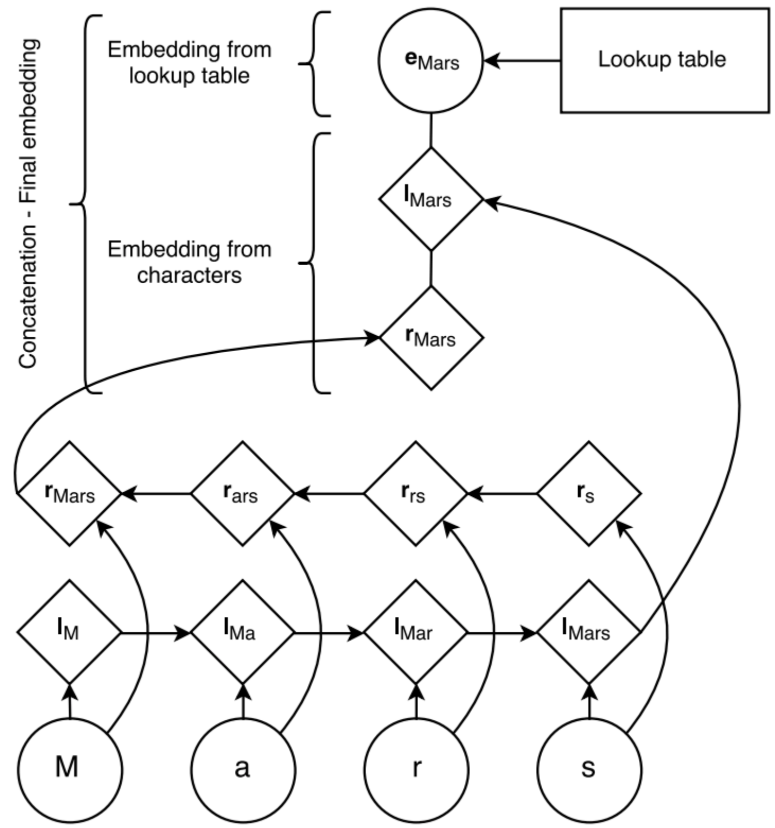
Model II: Chunking Algorithm

- Transition sequence example

Transition	Output	Stack	Buffer	Segment
REDUCE(LOC)	[(Mark Watney)-PER, visited, Mars-LOC]	[]	[]	Mars-LOC

Word Embeddings

- Character-based model of words
  - Character-level BiLSTM
- Pretrained embeddings
  - Skip-n-gram (Ling et al., 2015)
    - Word2vec that accounts for word order
    - Pretrained on
      - Spanish Gigaword version 3
      - Leipzig corpora collection
      - German monolingual data from 2010 WMT
      - English Gigaword version 4



Training

- Neural Network training
  - Back-propagation
  - SGD with gradient clipping
- Hyperparameters
  - LSTM dimension: 100
  - Dropout rate: 0.5
  - Embedding for transition: 16

Results

- Experiment on English
  - CoNLL-2003

Model	**F_1**
Collobert et al. (2011)*	89.59
Lin and Wu (2009)	83.78
Lin and Wu (2009)*	90.90
Huang et al. (2015)*	90.10
Passos et al. (2014)	90.05
Passos et al. (2014)*	90.90
Luo et al. (2015)* + gaz	89.9
Luo et al. (2015)* + gaz + linking	**91.2**
Chiu and Nichols (2015)	90.69
Chiu and Nichols (2015)*	90.77
LSTM-CRF (no char)	90.20
LSTM-CRF	**90.94**
S-LSTM (no char)	87.96
S-LSTM	90.33

Table 1: English NER results (CoNLL-2003 test set). * indicates models trained with the use of external labeled data

Results

- Experiment on German
  - CoNLL-2003

Model	$\mathbf{F_1}$
Florian et al. (2003)*	72.41
Ando and Zhang (2005a)	75.27
Qi et al. (2009)	75.72
Gillick et al. (2015)	72.08
Gillick et al. (2015)*	76.22
LSTM-CRF – no char	75.06
LSTM-CRF	**78.76**
S-LSTM – no char	65.87
S-LSTM	75.66

Table 2: German NER results (CoNLL-2003 test set). * indicates models trained with the use of external labeled data

Results

- Experiment on Spanish
  - CoNLL-2002

Model	$\mathbf{F_1}$
Carreras et al. (2002)*	81.39
Santos and Guimarães (2015)	82.21
Gillick et al. (2015)	81.83
Gillick et al. (2015)*	82.95
LSTM-CRF – no char	83.44
LSTM-CRF	**85.75**
S-LSTM – no char	79.46
S-LSTM	83.93

Table 4: Spanish NER (CoNLL-2002 test set). * indicates models trained with the use of external labeled data

Results

- Experiment on Dutch
  - CoNLL-2002

Model	$\mathbf{F_1}$
Carreras et al. (2002)	77.05
Nothman et al. (2013)	78.6
Gillick et al. (2015)	78.08
Gillick et al. (2015)*	**82.84**
LSTM-CRF – no char	73.14
LSTM-CRF	**81.74**
S-LSTM – no char	69.90
S-LSTM	79.88

Table 3: Dutch NER (CoNLL-2002 test set). * indicates models trained with the use of external labeled data

Ablation

Model	Variant	F_1
LSTM	char + dropout + pretrain	89.15
LSTM-CRF	char + dropout	83.63
LSTM-CRF	pretrain	88.39
LSTM-CRF	pretrain + char	89.77
LSTM-CRF	pretrain + dropout	90.20
LSTM-CRF	pretrain + dropout + char	**90.94**
S-LSTM	char + dropout	80.88
S-LSTM	pretrain	86.67
S-LSTM	pretrain + char	89.32
S-LSTM	pretrain + dropout	87.96
S-LSTM	pretrain + dropout + char	90.33

Table 5: English NER results with our models, using different configurations. "pretrain" refers to models that include pretrained word embeddings, "char" refers to models that include character-based modeling of words, "dropout" refers to models that include dropout rate.

Conclusion

- Two neural architectures for sequence labeling
  - The best NER results in standard evaluation settings at the time of publish
  - Comparable performance with models that use external resouces
- Key aspects
  - Model output label dependencies
  - Word representations are crucial

DESIGN CHALLENGES AND MISCONCEPTIONS IN NEURAL SEQUENCE LABELING

By Jie Yang, Shuailong Liang, and Yue Zhang

Presented by Jamshed Kaikaus
CS 546 Spring 2020

Motivation

- Numerous state-of-the-art models on sequence labeling tasks (NER, Chunking, POS Tagging, etc.)
- However, reproducing published work can be challenging
- Why? Likely due to sensitivity on experimental settings and inconsistent configurations

Models	Word LSTM+CRF	Word LSTM	Word CNN+CRF	Word CNN
No Char	Huang et al. (2015)* Lample et al. (2016) Strubell et al. (2017)*	Ma and Hovy (2016) Strubell et al. (2017)*	Collobert et al. (2011)* dos Santos et al. (2015) Strubell et al. (2017)*	Strubell et al. (2017)*
Char LSTM	Lample et al. (2016) Rei (2017) Liu et al. (2018)	Lample et al. (2016)	No existing work	No existing work
Char CNN	Ma and Hovy (2016) Chiu and Nichols (2016)* Peters et al. (2017)	Ma and Hovy (2016)	dos Santos et al. (2015)	Santos and Zadrozny (2014)

Inconsistent Configurations pt. 1

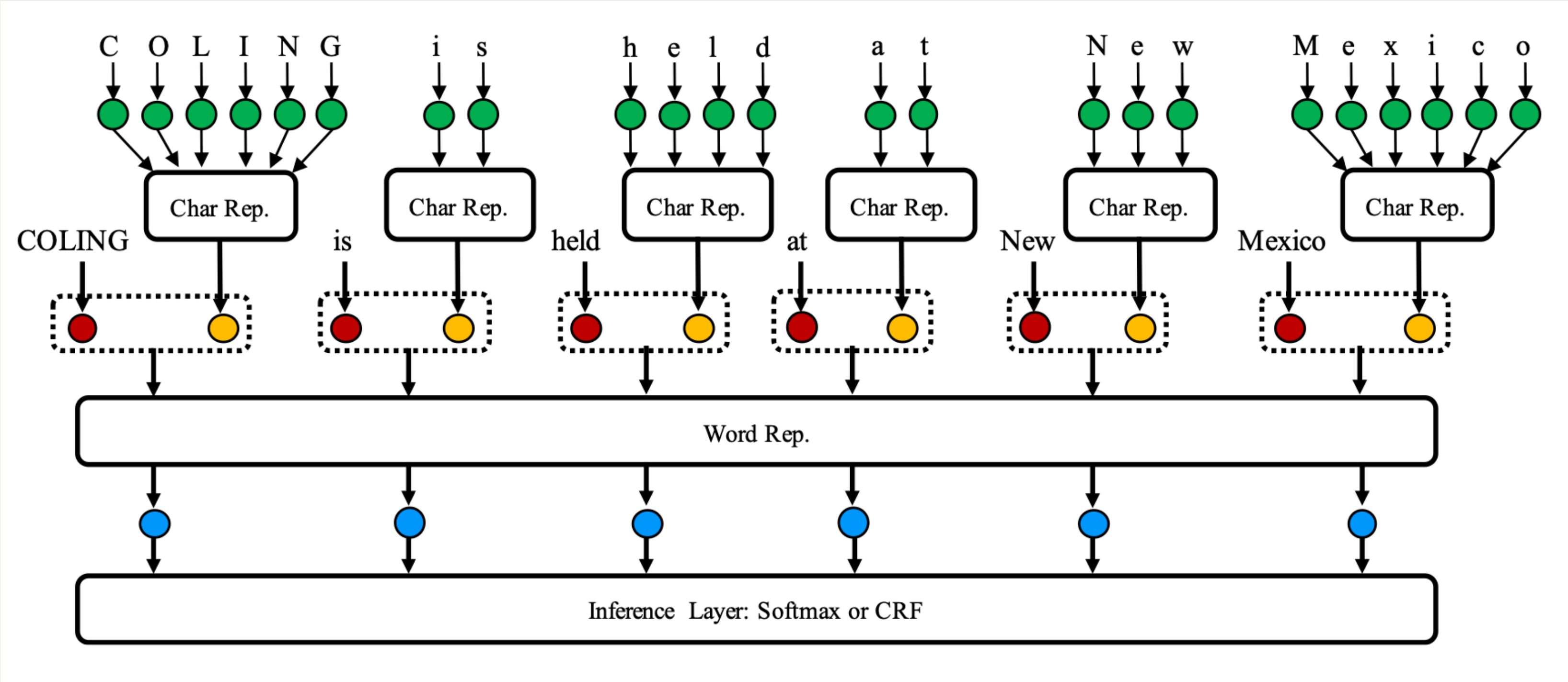
Datasets	CoNLL 2003 English NER	PTB POS	Combos?	Modifications?
Preprocessing	Normalize digit characters	Fine-grained Representations	None?	
Features	Word Spelling features	Context features	Neural features	No 'Hand-Crafted' features

Inconsistent Configurations pt. 2

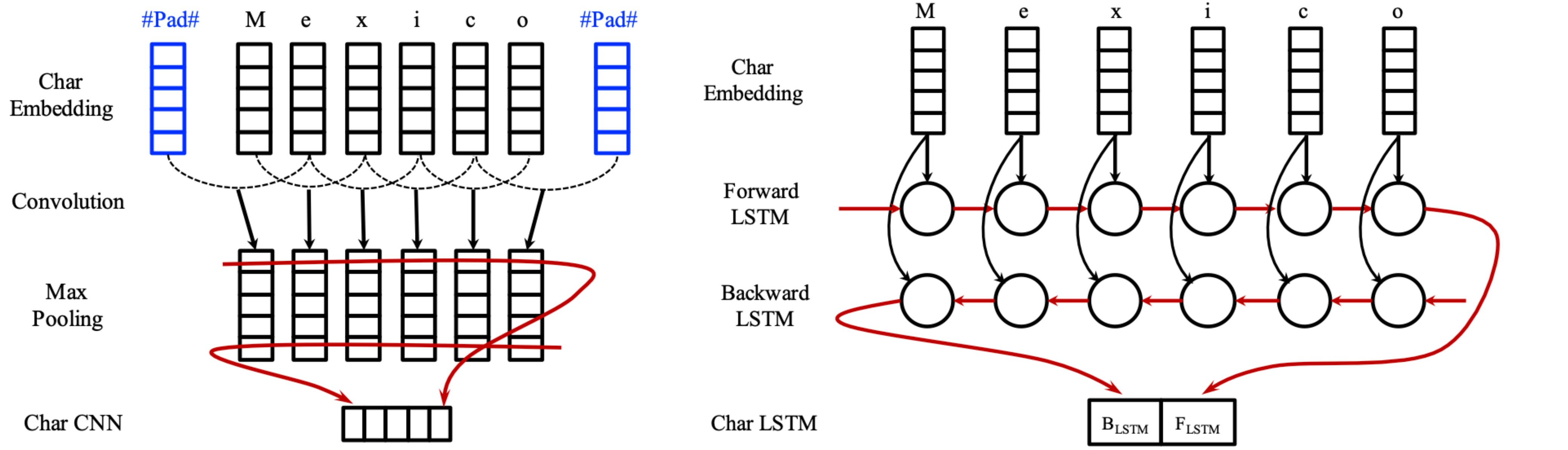
- Hyperparameters
  - Learning Rate
  - Dropout Rate
  - Num. Layers
  - etc.
- Evaluation
  - Mean + Std. Deviation, different random seeds
  - Best results among different trials
- Hardware
  - GPU
  - CPU

Proposal – A Unified Framework

- Authors implement a unified neural sequence labeling framework containing three layers:
  1. *Character Sequence Representation layer*
  2. *Word Sequence Representation layer*
  3. *Inference Layer*



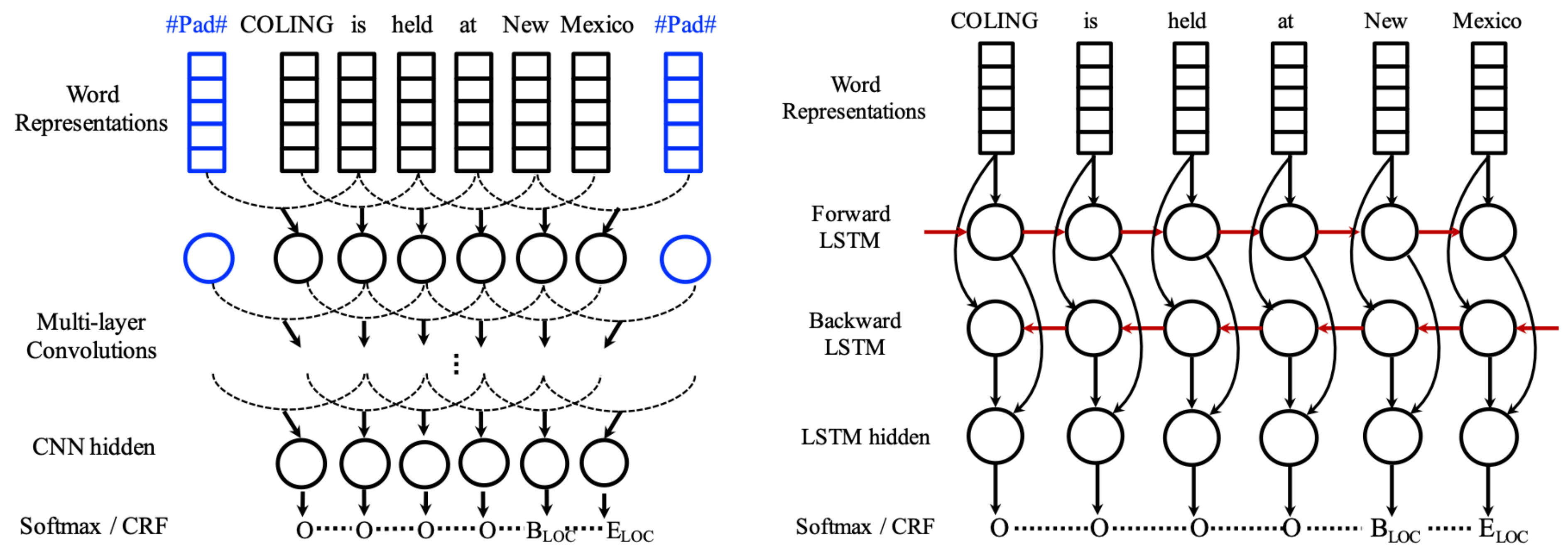
Character Sequence Representation Layer



(a) Character CNN.

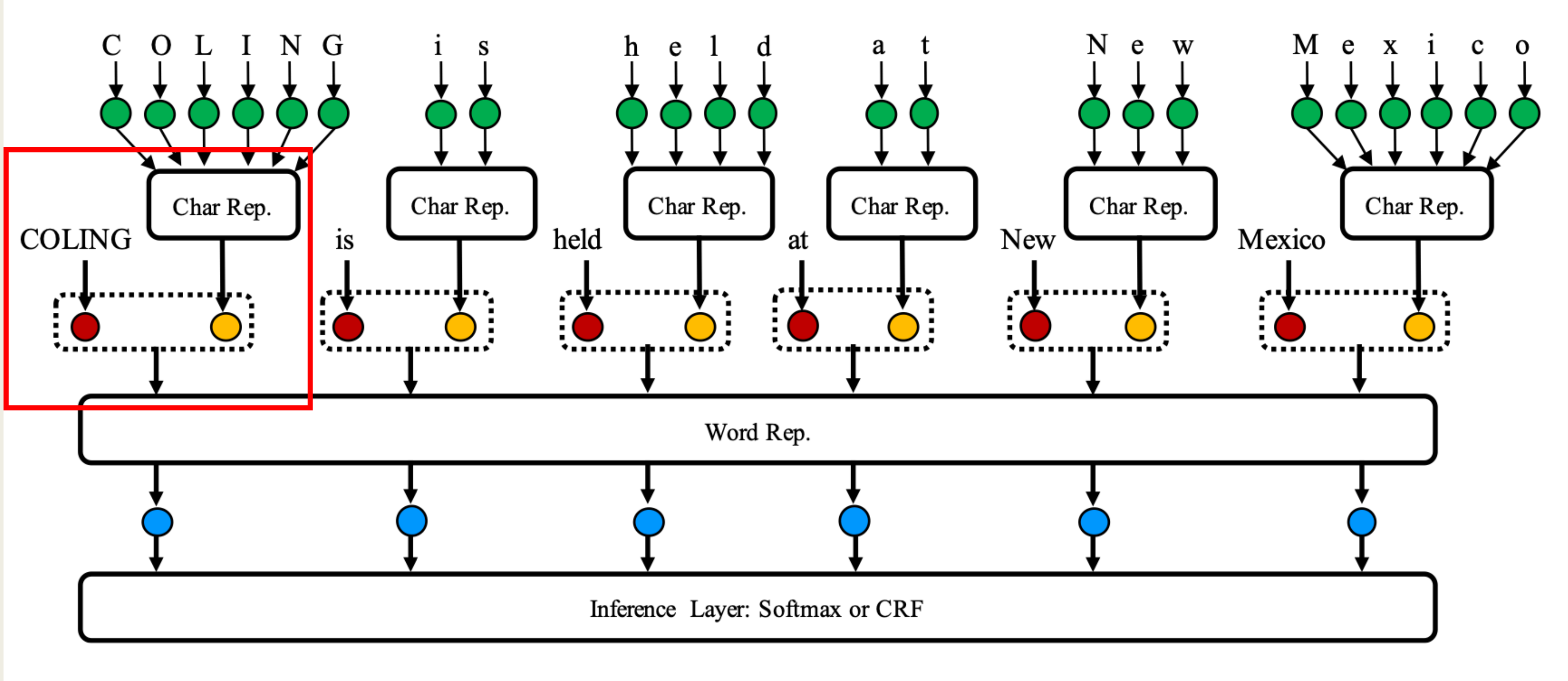
(b) Character LSTM.

Word Sequence Representation Layer

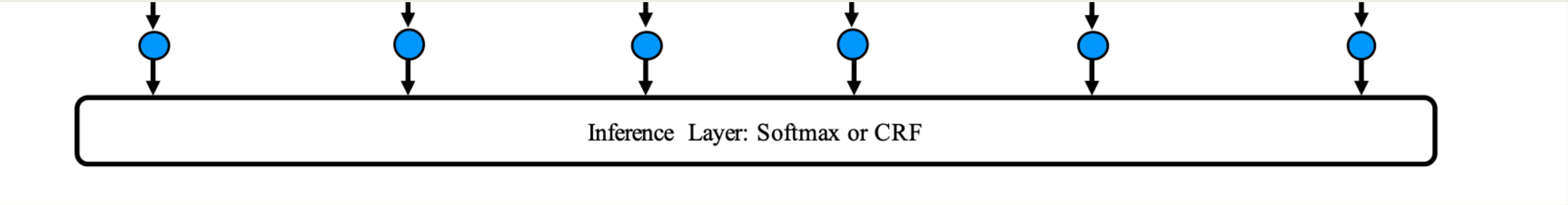


(a) Word CNN.

(b) Word LSTM.


C O L I N G
i s
h e l d
a t
N e w
M e x i c o
Char Rep.
Char Rep.
Char Rep.
Char Rep.
Char Rep.
Char Rep.
COLING
is
held
at
New
Mexico
Word Rep.
Inference Layer: Softmax or CRF

Inference Layer



- Takes output of previous layer (word sequence representations) as input
- Assigns labels to the word sequence as output
- Two options are examined as the inference layer:
  1. *Independent local decoding with a linear layer mapping WSR to label vocabulary, followed by* ***softmax***
  2. *Tasks with strong output label dependency,* ***CRF*** *is used*

Experimental Setup

- Three sequence labeling tasks to help comparison: **NER**, **Chunking**, and **POS Tagging**

	NER	Chunking	POS Tagging
Data	CoNLL 2003 English NER	CoNLL 2000 Shared Task	Peen Treebank – WSJ Portion
Evaluation	Precision	Precision	Token Accuracy
	Recall	Recall	
	F1-Score	F1-Score	

- Hyperparameters used include the following:
  - *Learning Rate* $(\eta_{LSTM} = 0.015, \eta_{CNN} = 0.005)$
  - *GloVe 100-dim used to initialize word embeddings; Character embeddings were randomly initialized*
  - *SGD with a decayed learning rate to update parameters*
  - *BIOES tag scheme for NER and Chunking*

Results – Named Entity Recognition

Results (F1-score)			NER			
			WLSTM+CRF	WLSTM	WCNN+CRF	WCNN
Nochar	Literature		90.10 (H-15)* 90.20 (L-16) 90.43 (S-17)*	87.00 (M-16) 89.34 (S-17)*	89.59 (C-11)* 90.54 (S-17)*	89.97 (S-17)*
	Ours	Max	89.45	88.57	88.90	88.56
		Mean±std	89.31±0.10	88.49±0.17	88.65±0.20	88.50±0.05
CLSTM	Literature		90.94 (L-16) 91.20 (Y-17)‡	89.15 (L-16)	–	–
	Ours	Max	91.20	90.84	90.70	90.46
		Mean±std	91.08±0.08	90.77±0.06	90.48±0.23	90.28±0.30
CCNN	Literature		90.91±0.20 (C-16) 91.21 (M-16) 90.87±0.13 (P-17)	89.36 (M-16)	–	–
	Ours	Max	91.35	90.73	90.43	90.51
		Mean±std	91.11±0.21	90.60±0.11	90.28±0.09	90.26±0.19

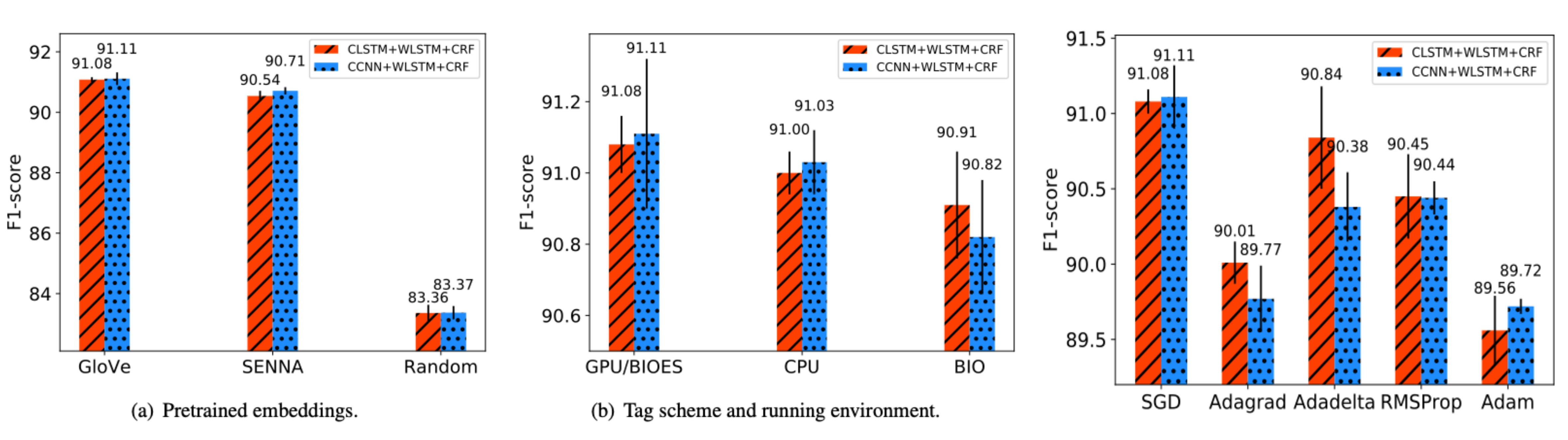
Results – Chunking

Results (F1-score)			chunking			
			WLSTM+CRF	WLSTM	WCNN+CRF	WCNN
Nochar	Literature		94.46 (H-15)*	94.13 (Z-17) 95.02 (H-17)*	94.32 (C-11)*	–
	Ours	Max	94.49	93.79	94.23	94.12
		Mean±std	94.37±0.11	93.75±0.04	94.11±0.08	94.08±0.06
CLSTM	Literature		93.15 (R-17) 94.66 (Y-17)‡	–	–	–
	Ours	Max	95.00	94.33	94.76	94.55
		Mean±std	94.93±0.05	94.28±0.04	94.66±0.01	94.48±0.07
CCNN	Literature		95.00±0.08 (P-17)	–	–	–
	Ours	Max	95.06	94.24	94.77	94.51
		Mean±std	94.86±0.14	94.19±0.04	94.66±0.13	94.47±0.03

Results – POS Tagging

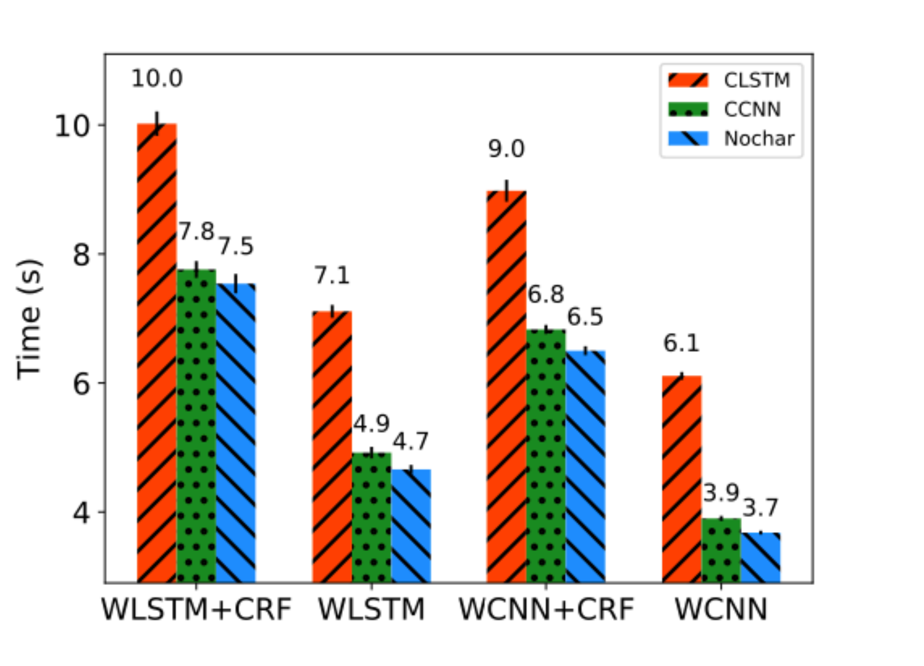
Results (Accuracy)			POS			
			WLSTM+CRF	WLSTM	WCNN+CRF	WCNN
Nochar	Literature		97.55 (H-15)*	96.93 (M-16) 97.45 (H-17)*	97.29 (C-11)*	96.13 (S-14)
	Ours	Max Mean±std	97.20 97.19±0.01	97.23 97.20±0.02	96.99 96.95±0.04	97.07 97.01±0.04
CLSTM	Literature		97.35±0.09 (L-16)† 97.55 (Y-17)‡	97.78 (L-15)	–	–
	Ours	Max Mean±std	97.49 97.47±0.02	97.51 97.48±0.02	97.38 97.33±0.03	97.38 97.33±0.04
CCNN	Literature		97.55 (M-16)	97.33 (M-16)	–	97.32 (S-14)
	Ours	Max Mean±std	97.46 97.43±0.02	97.51 97.44±0.04	97.33 97.29±0.03	97.33 97.30±0.02

Results – External Factors



- Models using pre-trained embeddings show significant improvements
- Models using BIOES tag schemes perform significantly better than those that use BIO
- SGD outperforms all other optimizers significantly

Analysis – Decoding Speed



- CRF Inference layer limits decoding speed due to the left-to-right inference process
- Char. LSTM significantly slows down the system
- Adding Char. CNN does not affect decoding speed but gives significant accuracy improvements
- Word-Based CNN are significantly faster than Word-Based LSTM, with close accuracies

Analysis – Out-Of-Vocabulary

Results	NER (F1-score)				chunking (F1-score)				POS (Accuracy)			
	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV
Nochar+WLSTM+CRF	91.33	87.36	100.00	69.68	94.87	90.84	95.51	91.47	97.51	89.76	94.07	75.36
CLSTM+WLSTM+CRF	**92.18**	90.63	100.00	78.57	**95.20**	**92.65**	94.38	**94.01**	**97.63**	**93.82**	94.07	**87.32**
CCNN+WLSTM+CRF	91.76	**91.25**	100.00	**81.58**	95.15	92.34	**97.75**	93.55	97.62	93.33	**94.69**	83.82
Nochar+WCNN+CRF	90.71	86.99	100.00	69.09	94.56	90.98	93.26	91.71	97.29	89.10	**94.17**	74.15
CLSTM+WCNN+CRF	**91.59**	90.07	100.00	77.92	**95.02**	91.86	94.38	**93.32**	**97.48**	**93.28**	**94.17**	**88.29**
CCNN+WCNN+CRF	91.35	**90.46**	100.00	**78.88**	94.83	**92.42**	**96.63**	92.40	97.46	92.74	93.86	87.80

- Char. LSTM or CNN representations improve OOTV and OOBV the most
  - Proves neural character sequence representations disambiguate the OOV words
- Char. LSTM representations give best IV scores across all configurations

Takeways

- Character information improves model performances
- LSTM vs. CNN
  - *Comparable improvements at the character-level*
  - *LSTM encoder provide better performance at the word-level*
  - *CNN generally more efficient*
- CRF Inference algorithm is effective on NER and chunking tasks
- BIOES tags are better than BIO
- Pretrained embeddings and SGD optimizer provide better performance