Review on NLP(Almost) from Scratch

Sunkyung Park

- **NLP is not my specialty. I have little experience with it.**
- **Although this paper covers the basics, it was too difficult for me.**
- **If any thing wrong, please let me know.**

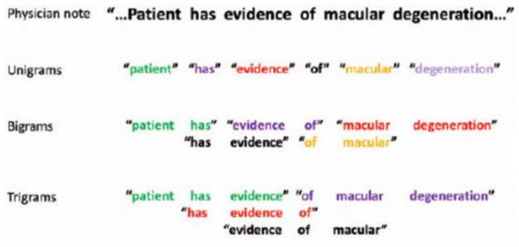
- **Outline**

Corpus → **Labeling** → **Vectorization (Word Embedding)** → **Dictionary** → **LM (Inference)**

2. Benchmark Tasks ➔ How to Label in Corpus?

2.1 Part-Of-Speech(POS) Tagging

- Label each word w/ syntactic role(plural noun, adverb)
- Best POS classifier : trained windows of text, which are the fed to a bidirectional decoding algorithm
- Features
  ① Preceding and following tag context
  ② Multiple words(bigrams, trigrams…)



  ③ Handcrafted features

2.2 Chunking

- Shallow parsing : *an analysis of a sentence which first identifies constituent parts of sentences (nouns, verbs, adjectives, etc.) and then links them to higher order units that have discrete grammatical meanings (noun groups or phrases, verb groups, etc.)*
- Each word assigned only one unique tag. e.g B-NP(Begin Noun Phrase), I-NP(inside Noun Phrase)
- Shen and Sarkar(2005): Voting classifier scheme
  ① Each classifier is trained on different tag representations(IOB, IOE)
  ② (+) Build trigrams with features
    a. POS features
    b. Hand-crafted specialization…
  ③ Viterbi decoding at test time

2.3 Name Entity Recognition(NER)

- Labels atomic elements in the sentence into categories “PERSON” or “LOCATION”
- (as in chunking task) each word assigned a tag prefixed by an indicator of beginning or the inside of an entity.
- Ando and Zhang(2005)
  ① A linear model on two auxiliary unsupervised tasks.
  ② Viterbi decoding.
  ③ Unlabeled corpus, 27M words from Reuters.
  ④ Features
    a. Words
    b. POS tags
    c. Suffixes(접미사), prefixes(접두사)
    d. CHUNKG tags

2.4 Semantic Role Labeling(SRL)

- Give a semantic role to a syntactic constituent of a sentence
- SRL systems w/ several stages
  ① Producing a parse tree
  ② Identifying which parse tree nodes represent a given verb(술부)
  ③ Classifying the nodes to compute (corresponding) the SRL tags.
- Feature categories(Gildea and Jurafsky, 2002; Pradhan et al., 2004)
  ① POS, syntactic labels, nodes in the tree
  ② node’s position(left of right) in verb
  ③ Whether a node in the parse tree is part of a noun or verb phrase

[Experimental environment setup]

Task	Benchmark	Data set	Training set (#tokens)	Test set (#tokens)	(#tags)
POS	Toutanova et al. (2003)	WSJ	sections 0–18 (912,344)	sections 22–24 (129,654)	(45)
Chunking	CoNLL 2000	WSJ	sections 15–18 (211,727)	section 20 (47,377)	(42) (IOBES)
NER	CoNLL 2003	Reuters	“eng.train” (203,621)	“eng.testb” (46,435)	(17) (IOBES)
SRL	CoNLL 2005	WSJ	sections 2–21 (950,028)	section 23 + 3 Brown sections (63,843)	(186) (IOBES)

* WSJ: Wall Street Journal

[Prior researches]

System	Accuracy
Shen et al. (2007)	97.33%
Toutanova et al. (2003)	97.24%
Giménez and Màrquez (2004)	97.16%

(a) POS

System	F1
Shen and Sarkar (2005)	95.23%
Sha and Pereira (2003)	94.29%
Kudo and Matsumoto (2001)	93.91%

(b) CHUNK

System	F1
Ando and Zhang (2005)	89.31%
Florian et al. (2003)	88.76%
Kudo and Matsumoto (2001)	88.31%

(c) NER

System	F1
Koomen et al. (2005)	77.92%
Pradhan et al. (2005)	77.30%
Haghighi et al. (2005)	77.04%

(d) SRL

* Accuracy: per-word accuracy

		Actual	
		True	False
Predicted	True	True Positive	False Positive
	False	False Negative	True Negative

- $Accuracy = \frac{True\ Positive + True\ Negative}{TruePostive + TrueNegative + FalsePositive + FalseNegative}$
- $F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$
- $Recall = \frac{True\ Positive}{TruePostive + FalseNegative}$
- $Precision = \frac{True\ Positive}{TruePostive + FalsePositive}$

3. Networks(for word embedding)

- Traditional NLP
  - ① Extract hand-designed features from sentence
  - ② Fed to Standard classification algorithm(e.g. SVM)
  - ③ The choice of features = empirical process(linguistic intuition)
  - ④ Task-dependent, computational cost ↑
- Newly Different approach
  - ① Pre-process features as little as possible
  - ② Use NN, trained in an end-to-end fashion
    - a. 1st Layer: Extracts features for each word(by labeling)
    - b. 2nd Layer: Extracts features from a window of words or from the whole sentence(treated as sequence)

3.1 Notations

$$f_\theta(\cdot) = f_\theta^L(f_\theta^{L-1}(\ldots f_\theta^1(\cdot)\ldots)).$$

- A Neural network with parameters θ
- Composition functions

$$[A]_{i,j}$$

- Coefficient at row i and column j.

$$\langle A \rangle_i^{d_{win}}$$

- Vector by concatenating the d_{win} column vectors around i^{th} column.

$$\langle A \rangle_i^1$$

- i^{th} column of Matrix *A*

, a sequence of element $\{x_1, x_2, \ldots, x_T\}$ is written $[x]_1^T$.

$$[x]_i$$

- i^{th} *element of the sequence*

$$\left[\langle A \rangle_i^{d_{win}}\right]^T = \left([A]_{1,i-d_{win}/2} \ldots [A]_{d_1,i-d_{win}/2}, \ldots, [A]_{1,i+d_{win}/2} \ldots [A]_{d_1,i+d_{win}/2}\right).$$

3.2 Transforming Words into Feature Vectors

- Ability to perform well with use of raw words(with enough training data, no need for preprocessing(lowercasing, encoding capitalization…) ➔ good word representation
- Words are fed to indices from a finite dictionary "$\mathcal{D}$"
- 1st Layer: map each word indices to a feature vector(with lookup table)
- Task of interest ➔ corresponding lookup table feature vector ➔ relevant word representation

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

- for each word, $w \in \mathcal{D}$, d_{wrd}-dimensional feature vector representation by lookup table $LT_w(\cdot)$
- *W* is a matrix of parameters d_{wrd} x $|\mathcal{D}|$
- $\langle W \rangle_w^1$ is w^{th} column of *W*.

🡐 Dictionary 형상과 똑같은 실수 Matrix에서 해당 단어의 위치 추출

- d_{wrd} is the word vector size.(hyper-parameter)

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

3.2.1 Extending to any discrete features

- A word as represented by *K* discrete features
  - ① e.g lower case stemmed root, lower case ending and capitalization
  - ② $w \in \mathcal{D}^1 \times \cdots \times \mathcal{D}^K$
  - ③ $\mathcal{D}^K$ is the dictionary for K^{th} feature

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

- A word w with K features, all lookup tables

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

- A sequence of words $|w|_1^T$

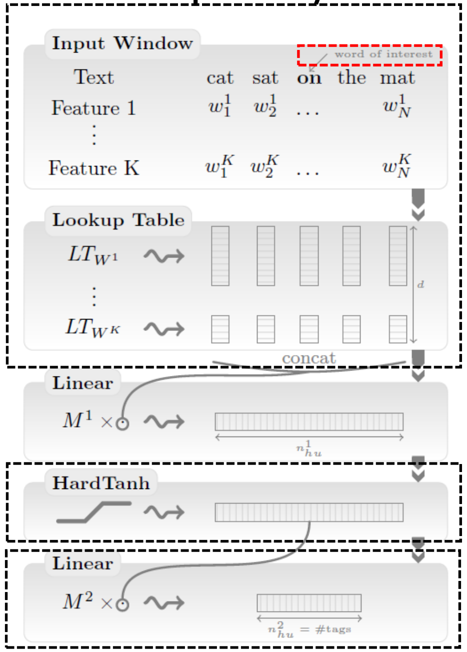
3. Networks

3.3 Extracting Higher Level Features from Word Feature

- Feature vectors produced by lookup table layer ➔ combined in subsequent layers ➔ a tag decision for each word
- Approaches which tag one word at the time
  ① A window approach
  ② (Convolutional) sentence approach

3.3.1 Window Approach(POS, CHUNKING, NER tasks)

- A tag of a word depends on neighboring words
- Process
  ① Given a word to tag
  ② a fixed size k_{sz} window of words around this word
  ③ The tag applies to the word located in the center of a window
  ④ Word features of fixed size d_{wrd} x k_{sz}
  ⑤ Each word in the window is passed through the lookup table layer.



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

- d_{wrd} = 1
- feature window of a sequence of words

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

- Output size = # of possible tags
- A score of the corresponding tag

Figure 1: Window approach network.

3.3.2 Sentence Approach(SRL task)

- Window approach, fails with SRL
  ① The tag of a word depends on a verb…what if the verb falls outside the window?
- Tagging a word requires the whole sentence

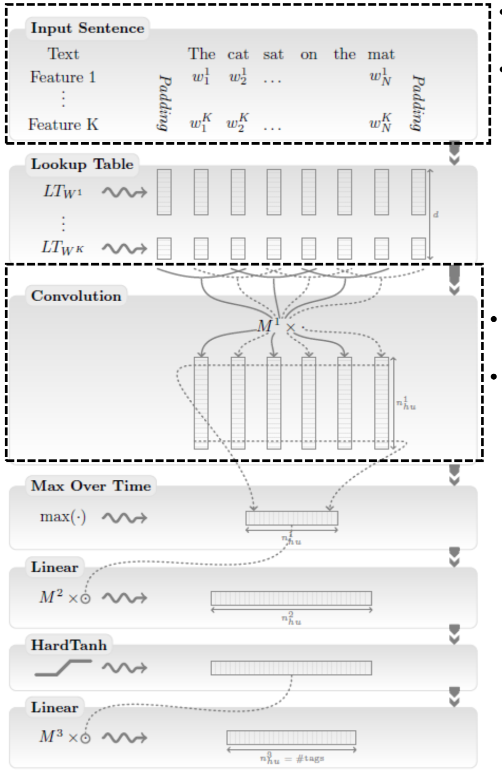
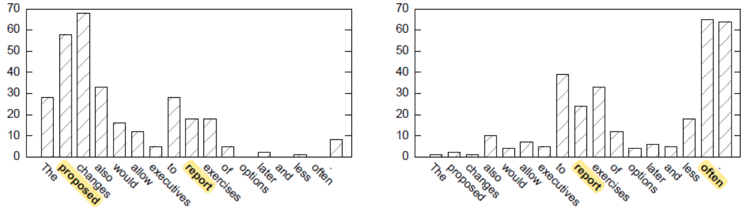


Figure 2: Sentence approach network.

- The complete sentence
- Given a word or a verb to tag ➔ each word augmented with two features
  - $i - pos_v$ (pos_v : the verb pos)
  - $i - pos_w$ (pos_w : the word pos)

- Extract local features around each word.
- Combines features into a global feature vector

Sentence Approach Results…



- The network catches features mostly around the verb of interest and word of interest
- Left: Proposed(word of interest) : Report(verb of interest)
- Right: Often(word of interest) : Report(verb of interest)

3.3.3 Tagging Scheme(Which format?)

- Labels with segments of a sentence
- Identify the segment boundaries

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

- This paper uses IOBES tagging scheme
- In the CHUNK task, noun phrases…
  ① S-NP: Mark a noun phrase containing a single word
  ② B-NP, I-NP, E-NP: first words, intermediate and last words of the noun phrase
  ③ O : words that not members of a chunk…

3. Networks

3.4 Training

- Two ways to interpret neural network outputs as probabilities
  ① Word-level log-likelihood
  ② Sentence-level log likelihood

3.4.1 WORD-LEVEL LOG-LIKELIHOOD

- Maximizing a likelihood over the training data.

$[f_\theta]_i$

- The NN with θ outputs a score, for i^{th} tag

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

- x : input example
- θ : parameters
- Normalization: Softmax operation over all the tags.

$$\log p(y\,|\,x,\theta) = [f_\theta]_y - \operatorname*{logadd}_j [f_\theta]_j \,.$$

- Cross entropy

3.4.2 SENETENCE-LEVEL LOGOLIKELIHOOD

- Considering correlation between the tag of a word in a sentence and its neighboring tags.
- Training scheme w/ sentence structure
  ① The predictions of all tags by our network for all words in a sentence.
  ② A score for going from one tag to another tag…
  ③ Valid paths of tags during training.

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

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

- The proposed NN results are behind the benchmark results, a vanilla neural network.

3. Networks

3.5 Supervised Benchmark Results

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

- Word embedding in the word lookup table of a Sematic Role Labeling
- Neighboring words in the embedding space do not seem to be semantically related.
- For the WSJ data, many words appear only a few times ➔ Difficult to train their feature vectors in the lookup table.
- Corpus 1 : Eng Wikipedia ➔ dataset w/ 631 million words
- Dictionary w/ 100,000

>>

4. Lots of Unlabeled Data

- Improve embeddings w/ large unlabeled data sets.
- Use this embeddings to initialize the work lookup tables

4.1 Data sets

- Corpus 1 : Eng Wikipedia ➔ dataset w/ 631 million words
- Corpus 2 : Reuters RCV1 ➔ dataset w/ extra 221 million words
- Dataset w/ 852 million words
- Dictionary w/ 130,000

4. Lots of Unlabeled Data

4.2 Ranking Criterion vs. Entropy Criterion

- Use unlabeled dataset to train language models
- It compute scores saying the acceptability of a piece of text = large neural networks w/ window approach
- [Entropy Criterion(maximizing)]
  ① Dictionary is large, computing the normalization can be extremely demanding.
  ② The entropy criterion largely determined by the most frequent phrases.
  ③ Rare but legal phrases are no less significant than common phrase.
- [Ranking Criterion(minimizing)]
  ① Compute a higher score when given a legal phrase than when given an incorrect phrase
  ② Pairwise ranking approach

$$\theta \mapsto \underbrace{\sum_{x\in\mathcal{X}} \sum_{w\in\mathcal{D}}}_{(a)} \max\left\{0\,,\,1-\underbrace{f_\theta(x)}_{(b)\ \uparrow}+\underbrace{f_\theta(x^{(w)})}_{(c)\ \downarrow}\right\},$$

(*) for minimization

(a):

X = all possible text windows with words in training corpus

D = dictionary of words

(b):

f : Score when applying all possible text windows

1-f : Loss ➔ loss for rare but legal phrase

(c): Score w/ text window in which the central word of the text window is replaced with words from the dictionary

4.3 Training Language Models

- Trained
  ① by stochastic gradient minimization of the ranking criterion,
  ② Sampling a sentence-word pair at each iteration
- Model selection(w/ hyperparameters)
  ① Breeding:
    a. Child networks initialized with the embeddings of their parents w/ different parameters
    b. Steps
      1) k processors, k initial parameter choices
      2) Choose another set of K parameters in the values from successful candidates from previous round(w/ the value of the ranking criterion) k parameters
- LM1
  ① Window size: 11
  ② Hidden layer: 100 units
  ③ Trained on en corpus(Wikipedia) w/ successive dictionaries(5,000 , 10,000, 30,000, 50,000 and 100,000 most common WSJ words)
- LM2
  ① Initialized with the embeddings of LM1

4.4 Embeddings(generalization performance on all tasks)

(old) Word embeddings in lookup table
In vanilla NN trained w/ dictionary sized 100,000

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

(new) Word embeddings in lookup table
In LM1 trained w/ dictionary sized 100,000 ➔ more satisfactory

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

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

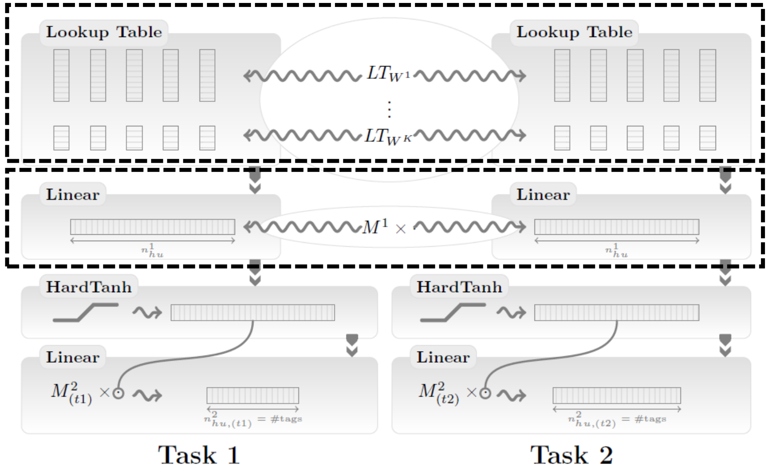
5. Multi-Task Learning

5.1 Joint Decoding vs. Joint Training

- [Joint Decoding]
  ① Combine the outputs of independently trained models
- [Joint Training]
  ① Joint decoding works w/ probabilistic dependency paths between models
  ② Joint training = an implicit supermodel ➔ discovering common internal representations.
  ③ It works when the training sets for each task have the same patterns with different labels.

5.2 Multi-Task Benchmark Results

- Training achieved by minimizing the loss averaged across all tasks.
- Stochastic gradient
  ① Picks examples for each task
  ② Applies to all the parameters of the corresponding model and shared parameters
  ➔ Equal weight to each task
- In this paper, fortunately, none of the training and test sets overlap across tasks ➔ It is necessary to check and remove the overlapping part first.



- Parameters of lookup tables and first hidden layer(linear layer for window approach, convolution layer for sentence approach) are shared
- Last layer is task-specific

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

- Results obtained by jointly trained models
- In POS, CHUNK, SRL w/ sentence Approach…
  ① without MTL(Multi-Task Learning) < w/ MTL
  ➔ Very little improvements

6. Temptation

- Explore what happens when increasing the level of task-specific engineering.

6.1 Suffix Features

- In POS task, discrete word features presenting the last two characters of every word
- Suffix dictionary size, 455
- Small improvement to benchmark system.

6.2 Gazetteers

- For NER systems use large dictionary w/ NER.
- Gazetteer by CoNLL challenge w/ 8,000 1) locations, 2) person names, 3) organizations, and 4) miscellaneous entities + on, odd(whether the word is in the gazetteer)
- This paper says that the large boost in performance is due to that gazetteers by CoNLL include word chunks in training set, not by language model.

6.3 Cascading

- Tags obtained for one task can be useful for taking decisions in other tasks.
- There is an improvement both the CHUNK and NER tasks with word features representing the POS tags.
- Same as SRL with word features representing the CHUNK tags.

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

6. Temptation

- Explore what happens when increasing the level of task-specific engineering.

6.4 Ensembles

- NN is non-convex, training runs w/ different initial parameters give different solutions
- [Voting ensemble]
  ① Voting 10 network outputs on a per tag basis > the average network performance
- [Joined ensemble]
  ① 10 network output scores combined w/ additional linear layer
  ② Fed to a new sentence-level likelihood

Approach		POS (PWA)	CHUNK (F1)	NER (F1)
Benchmark Systems		97.24	94.29	89.31
NN+SLL+LM2+POS	worst	97.29	93.99	89.35
NN+SLL+LM2+POS	mean **of 10 runs**	97.31	94.17	89.65
NN+SLL+LM2+POS	best	97.35	94.32	89.86
NN+SLL+LM2+POS	voting ensemble	97.37	94.34	89.70
NN+SLL+LM2+POS	joined ensemble	97.30	94.35	89.67

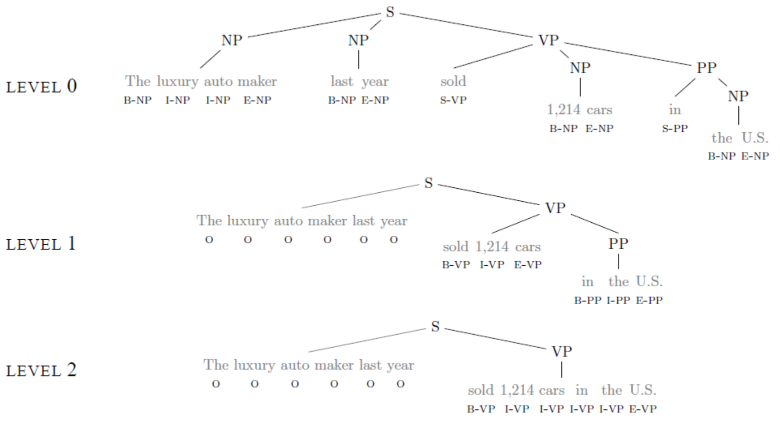
- Performance of networks obtained using by combining ten training runs w/ different initialization

6.5 Parsing

- Charniak parse tree by CoNLL 2005 data.
- Parse tree : A node in a syntactic parse tree assigns a label to a segment of parsed sentence.
- Application:
  ① Feed this labeled segmentation to proposed network through additional lookup tables.
  ② Each lookup table encode labeled segments of each parse tree level.
  ③ Labeled segments are fed to the network following a IOBES tagging scheme.

Approach	SRL (valid)	SRL (test)	
Benchmark System (six parse trees)	77.35	77.92	
Benchmark System (top Charniak parse tree only)	74.76	–	
NN+SLL+LM2	72.29	74.15	1.5%p
NN+SLL+LM2+Charniak (level 0 only)	74.44	75.65	0.16%p
NN+SLL+LM2+Charniak (levels 0 & 1)	74.50	75.81	0.24%p
NN+SLL+LM2+Charniak (levels 0 to 2)	75.09	76.05	-0.16%p
NN+SLL+LM2+Charniak (levels 0 to 3)	75.12	75.89	0.17%p
NN+SLL+LM2+Charniak (levels 0 to 4)	75.42	76.06	
NN+SLL+LM2+CHUNK	–	74.72	

- LM2 + additional lookup tables of dimension 5 for each parse tree level
- Performance: CHUNK < Parse Tree(Level 0 only)
  ➔ The latter identify leaf sentence segments that are often smaller than those by chunking tags.



- LEVEL 0 : info associated with leaves of the original Charniak parse tree
- LEVEL 1 to 4 : By repeatedly trimming the leaves
- "0" words belonging to the root node "S".

End of Document