Named Entity Recognition Using Deep Learning

Rudra Murthy

Center for Indian Language Technology,
Indian Institute of Technology Bombay

rudra@cse.iitb.ac.in

https://www.cse.iitb.ac.in/~rudra

Deep Learning Tutorial.
ICON 2017, Kolkata
21th December 2017

Outline

- What is NER?
- Traditional ML Approaches
- Motivating Deep Learning
- Deep Learning Solutions
- Summary

Introduction

What is Named Entity Recognition?

- The task of identifying **person** names, **location** names, **organization** names and other **miscellaneous entities** in a given piece of text.
- Example:
  - **Malinga** omitted from squad for **Pakistan** ODIs

  Malinga will be tagged as **Person** and **Pakistan** as **Location** entity

You thought NER was trivial

Challenges

- Named Entities are ambiguous
    - I went to Washington
    - I met Washington
- Named Entities form an open class
    - Box8
    - Alphabet
    - .
    - .

Challenges

List of Unique/Crazy Person Names

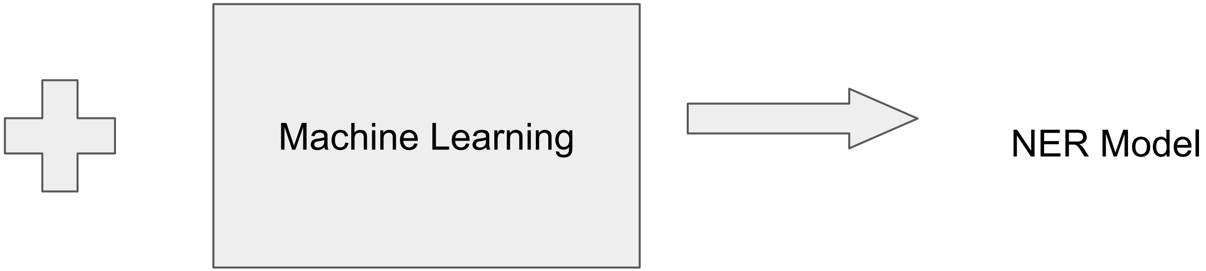
- Ahmiracle
- Anna…
- I'munique
- Baby Girl
- Abcde
- North West
- Melanomia
- Heaven Lee
- Tu Morrow
- Moxie Crimefighter
- Abstinence

- Apple
- Facebook
- Danger
- Colon
- Mercury Constellation Starcuiser
- Pilot Inspektor
- Rage
- Billion
- Audio Science
- Sadman
- Hashtag

Source: http://www.momjunction.com/articles/worst-baby-names-in-the-world_00400377/#gref

Traditional ML Approaches

Vince’s	Person
maiden	O
test	O
fifty	O
keeps	O
England	Misc
ticking	O
Mumbai	Misc
drop	O
Nayar	Person
.	.



Vince’s	Person
maiden	O
test	O
fifty	O
keeps	O
England	Team
ticking	O
Mumbai	Team
drop	O
Nayar	Person
.	.

+

Machine Learning
*learn probabilities over words

→ NER Model

P(Person | Vince’s) = ?
P(Location | Vince’s) = ?
P(Team | Vince’s) = ?
P(O | Vince’s) = ?

Problem Formulation

Given a word sequence ($w_1, w_2, \ldots, w_n$) find the most probable tag sequence ($y_1, y_2, \ldots, y_n$)

i.e, find the most probable entity label for every word in the sentence

Best Tag Sequence

$$y^* = P(\ y_1, y_2, \ldots, y_n \mid w_1, w_2, \ldots, w_n\ )$$

Why sequence labeling and not classification task?

Sequence labeling performs better at identifying named entity phrases

Problem Formulation (CRF)

Given a word sequence (w_1 , w_2 , … , w_n) find the most probable tag sequence (y_1 , y_2 , … , y_n)

$$P(\mathbf{y} \mid \mathbf{w}) = \exp\left(\Sigma_{i=1}^{n} \Sigma_k \lambda_k f_k (y_t , y_{t-1} , \mathbf{x}) \right)$$

Here, $f_k (y_t , y_{t-1} , \mathbf{x})$ is a feature function whose weights λ_k needs to be learned during training

The feature function is used to define various features

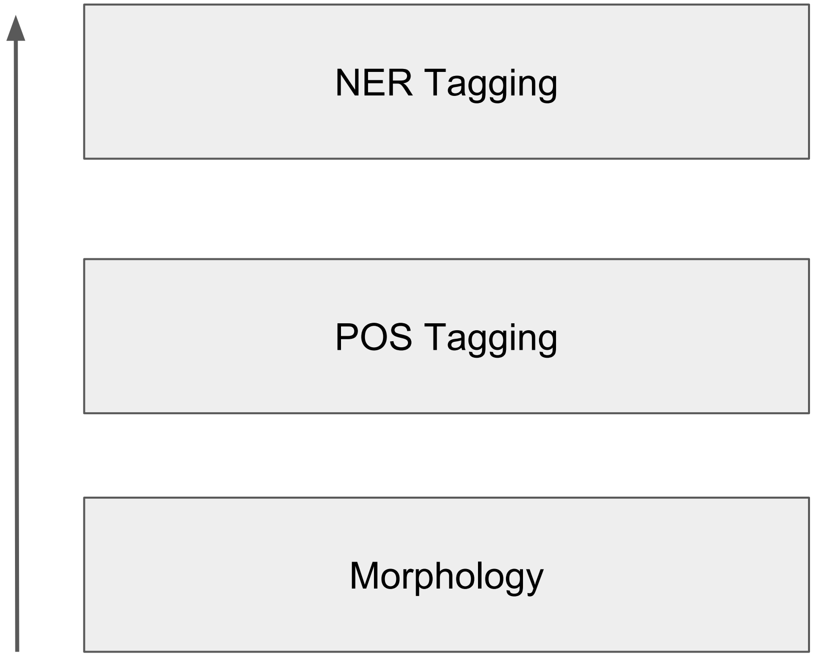
Typical Features

- **Word Features**
- **Subword Features**
- **Context Words**
- POS Tag
- Gazetteers
- Suffix Gazetteers
- Handcrafted Features
  - Does the word begin with an uppercase character?
  - Contains any digits?
  - Contains special characters?

Why Deep Learning?

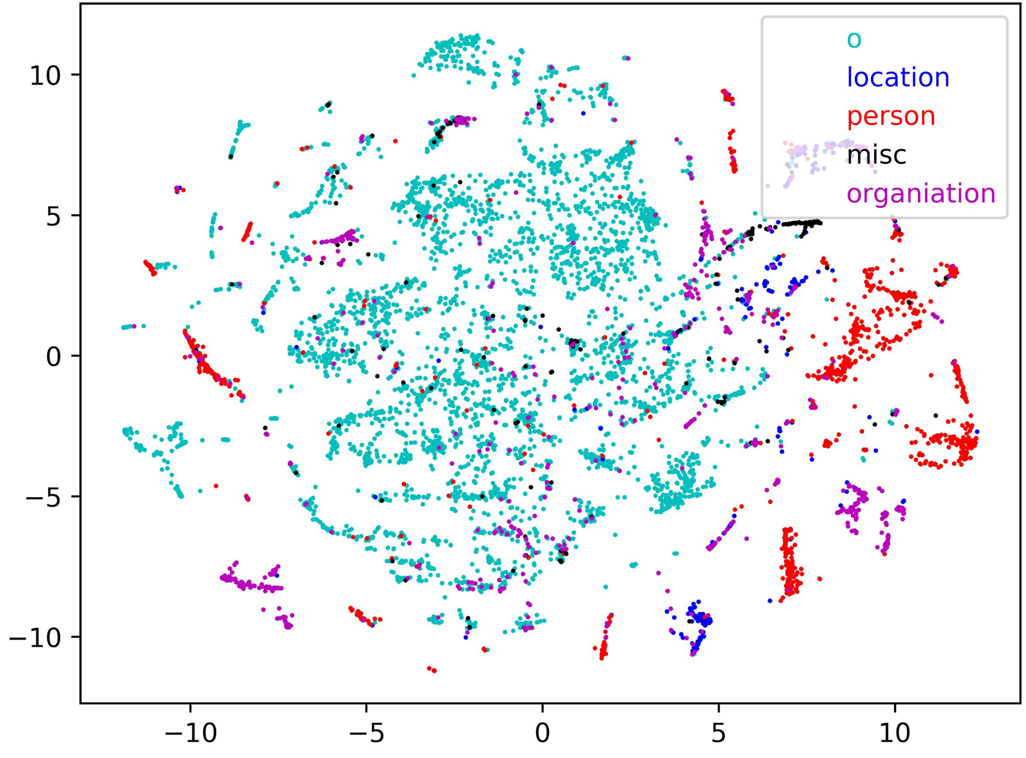
Why Deep Learning?

- Neural networks provide an hierarchical architecture
- Lower layers of the network can discover subword features
- Layers above it can be used to discuss word specific features
- The higher layer can use the information coming from lower layers to identify named entities



Word Embeddings

2-d plot of word embeddings annotated with named entity tags



- Plot of word Spectral word embedding for words from English CoNLL 2003 test data
- Choose the most frequent named tag for every word
- We observe named entities of the same type forming a cluster in the embedding space

Deep Learning Solutions

- We have looked at various neural network architectures
- What are the important features for NER?
- What neural network architectures can we use to make the model learn these features?

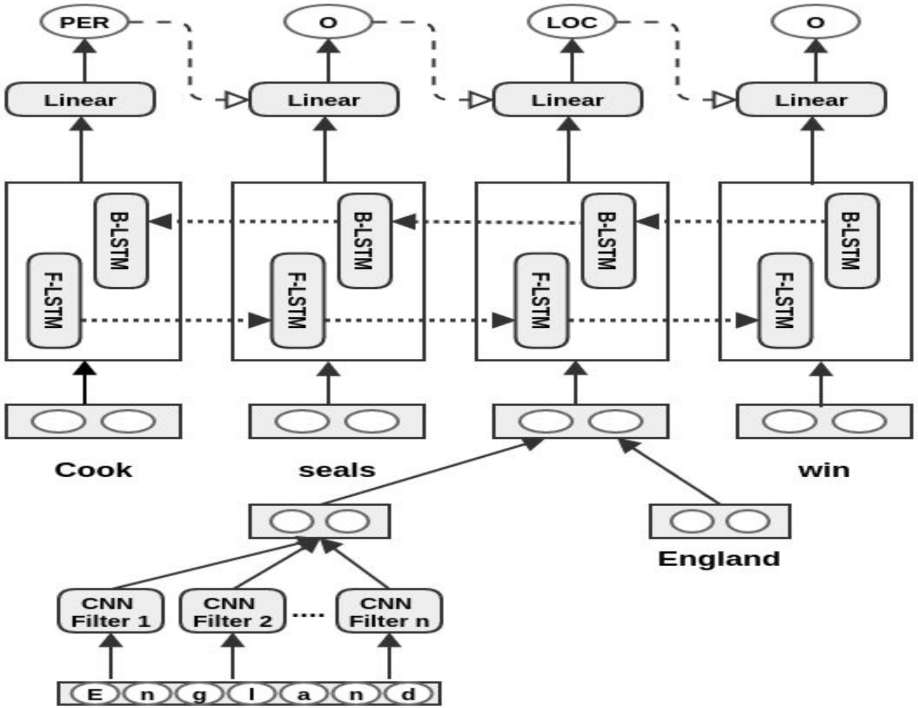
Deep Learning Model for NER Timeline

Model	Subword		Word	
	CNN	Bi-LSTM	CNN	Bi-LSTM
Hammerton [2003]				✔
Collobert et al. [2011]			✔	
dos Santos et al. [2015]	✔		✔	
Huang et al. [2015]				✔
Chiu and Nichols [2016]	✔			✔
Murthy and Bhattacharyya [2016]	✔			✔
Lample et al. [2016]		✔		✔
Ma and Hovy [2016]	✔			✔
Yang et al. [2017]		✔		✔

Deep Learning Model for NER [Murthy and Bhattacharyya [2016]]

- Given a dataset *D* consisting of tagged sentences
  - Let X = $\{x_1, x_2, \dots, x_n\}$ be the sequence of words in a sentence
  - Let Y = $\{y_1, y_2, \dots, y_n\}$ be the sequence of corresponding tags
- Goal is to maximize the likelihood of the tag sequence given the word sequence
  - maximize $P(Y \mid X)$
  - maximize $P(y_1, y_2, \dots, y_n \mid x_1, x_2, \dots, x_n)$
  - maximize $\Pi_{i=1}^{n} P(y_i \mid x_1, x_2, \dots, x_n\ y_{i-1})$
- We maximize the log-likelihood for every tag sequence in the training data

Deep Learning Architecture for NER



Deep Learning Architecture for NER

- The input to the model is words and the character sequence forming the word
- One-hot representation of the word is sent through a Lookup Table
- Lookup Table is initialized with pre-trained embeddings
- Additionally character sequence is fed to CNN to extract sub-word features
- The word embeddings and sub-word features are concatenated to get final word representation
- This representation is fed to a Bi-LSTM layer which disambiguates the word (w.r.t NER task) in the sentence
- Finally, the output from Bi-LSTM model is fed to softmax layer which predicts the named entity label

Word Embeddings

- Word embeddings represent words using d-dimensional real valued vector
- Word embeddings exhibit the property that named entities tend to form a cluster in the embedding space
- Providing word embedding features as input is more informative compared to the one-hot representation
- Word embeddings are updated during training

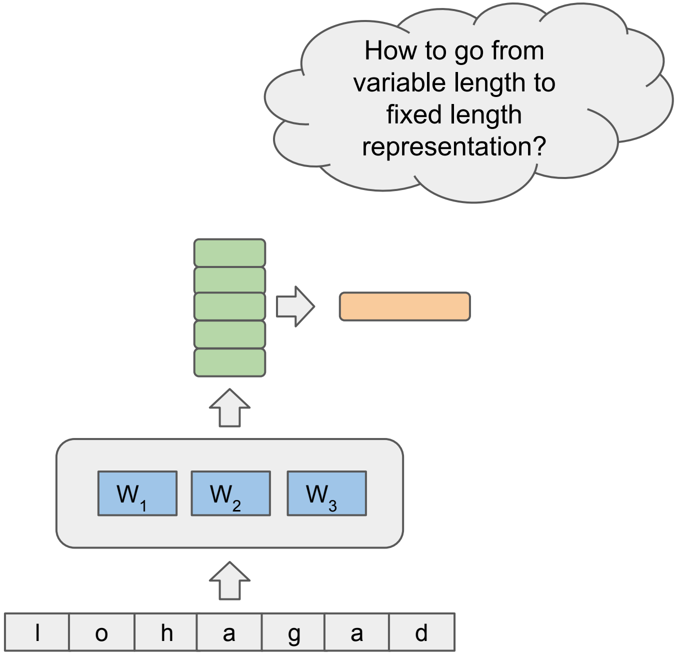
Subword Features

- We use multiple CNNs of varying width to extract sub-word features
- Every character is represented using one-hot vector representation
- The input is a matrix with i^{th} row indicating the one-hot vector of i^{th} character in the word
- The output of CNN is fed to max-pooling layer
- We extract 15-50 features from the CNN for every word
- This forms the sub-word features for the word

Subword Features

- This module should be able to discover various subword features
- The feature could be capitalization feature, affix features, presence of digits *etc.*

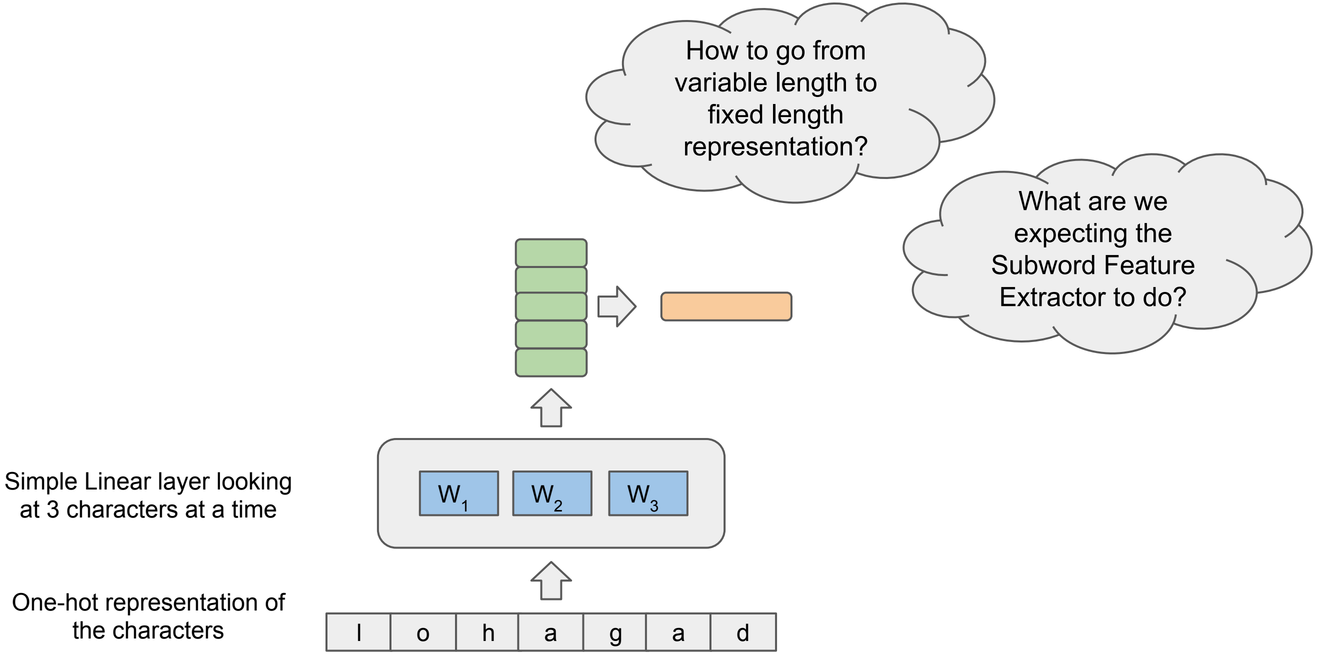
CNNs to Extract Subword Features



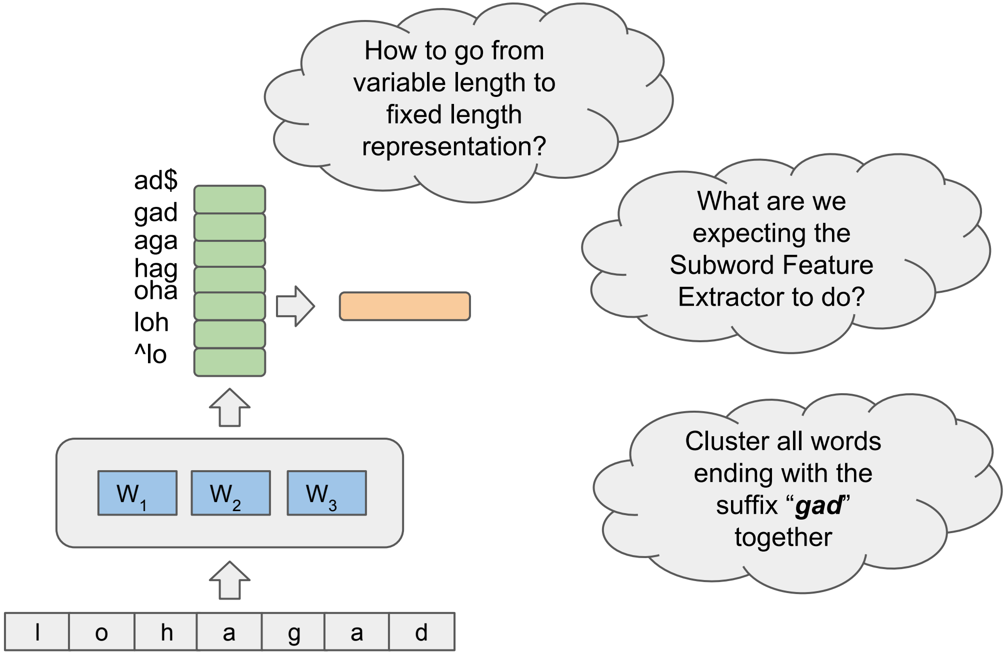
Simple Linear layer looking at 3 characters at a time

One-hot representation of the characters

CNNs to Extract Subword Features



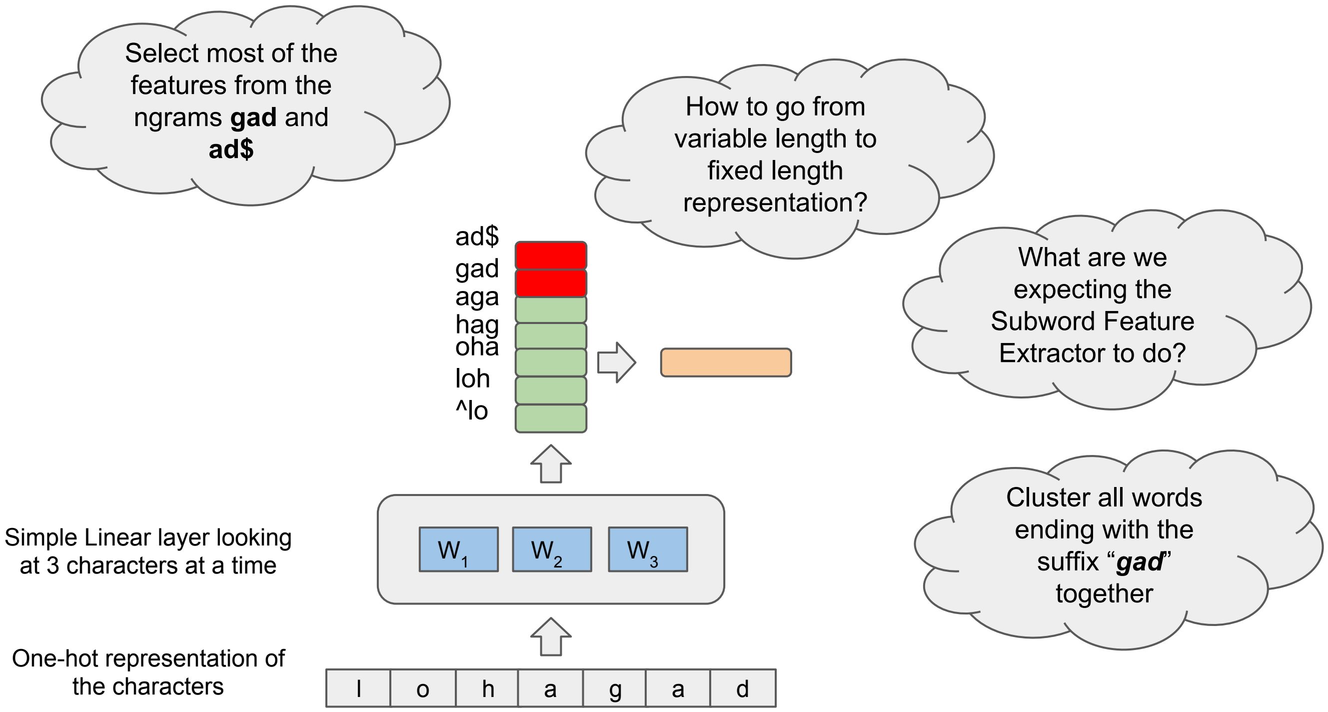
CNNs to Extract Subword Features



Simple Linear layer looking at 3 characters at a time

One-hot representation of the characters

CNNs to Extract Subword Features



CNNs to Extract Subword Features

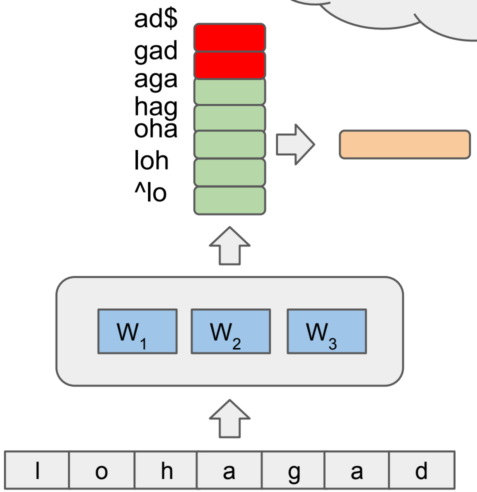
Use Max-Pooling

Select most of the features from the ngrams **gad** and **ad$**

How to go from variable length to fixed length representation?

What are we expecting the Subword Feature Extractor to do?

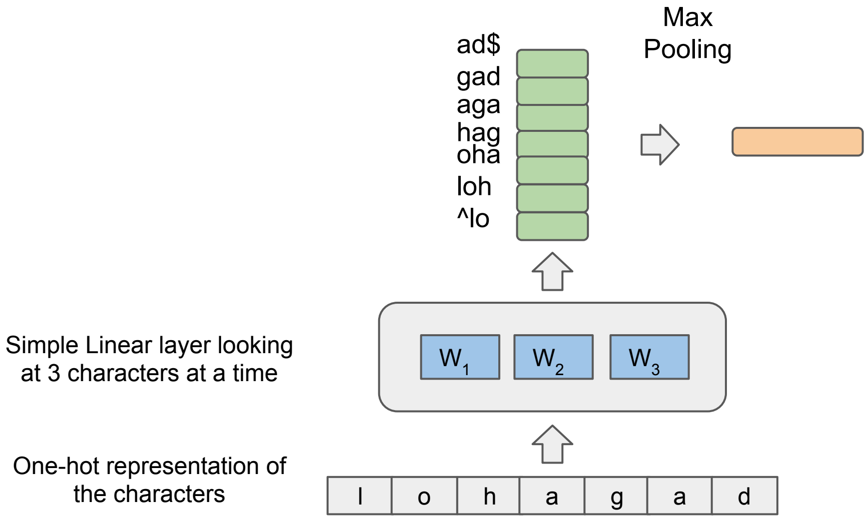
Cluster all words ending with the suffix “***gad***” together



Simple Linear layer looking at 3 characters at a time

One-hot representation of the characters

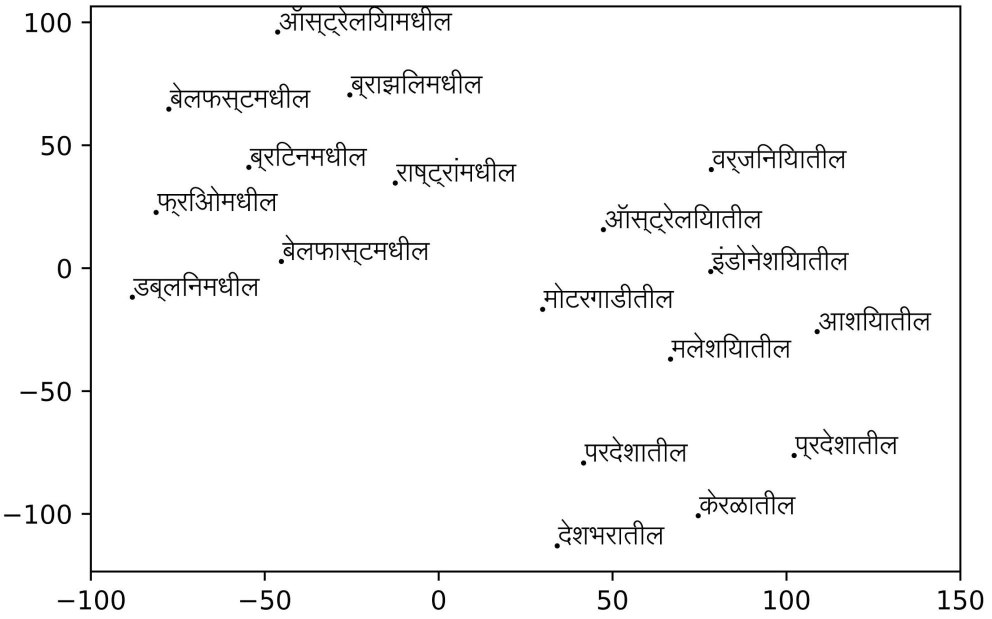
CNNs to Extract Subword Features



CNNs to Extract Subword Features

- Use CNNs to extract various sub-word features
- By extracting, we mean word with similar features to be closer in the feature space
- The features could be *capitalization features, similar suffixes, similar prefixes, all time expressions*, etc.
- This is similar to say suffix embeddings except that the suffix pattern is discovered by the model

Subword Features



- Plot of subword features for different Marathi words
- We observe that the CNN was able to cluster words based on their suffixes
- The CNN model was able to cluster words with similar suffixes

Bi-LSTM Layer

- We have observed that both word embeddings and subword features are able to cluster similar words together
- All location names forming a cluster in word embedding space
- All words with similar suffixes forming a cluster in the sub-word feature space
- This acts as a proxy feature for suffix features used in traditional ML methods
- Till now we have looked at only global features
- What about local features like contextual features?

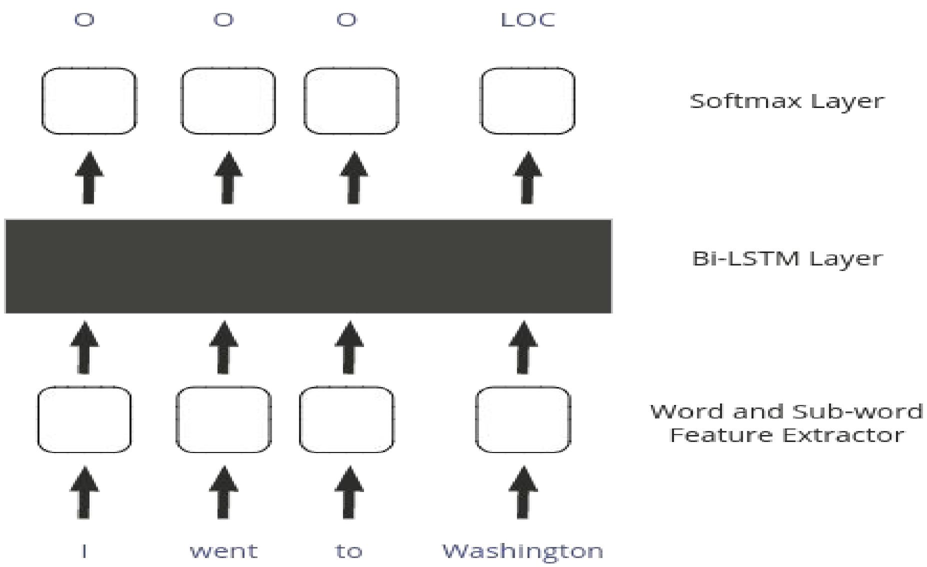
Bi-LSTM Layer

- The word embeddings and extracted sub-word features give global information about the word
- Whether a word is named entity or not depends on the specific context in which it is used
- For example,
  - I went to Washington
  - I met Washington
- The Bi-LSTM layer is responsible for disambiguation of the word in a sentence
- Here the disambiguation is w.r.t named entity tags

Bi-LSTM Layer

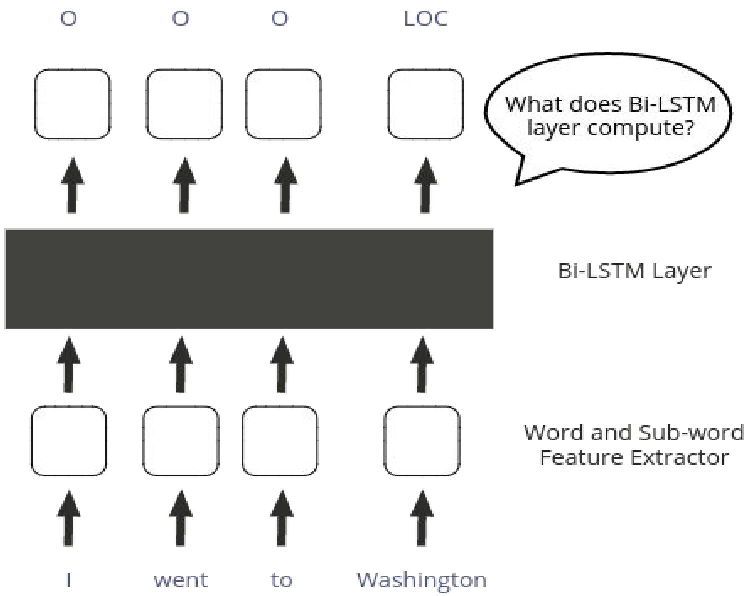
- Given a sequence of words, $\{x_1, x_2, \dots, x_n\}$ the Bi-LSTM layer employs two LSTM modules
- The forward LSTM module reads a sequence from left-to-right and disambiguates the word based on left context
- The forward LSTM extracts a set of features based on current word representation and previous word's forward LSTM output
  - $h_f^i = f(x_i, h_f^{i-1})$
- Similarly, backward LSTM reads sequence from right to left and disambiguates the word based on right context
  - $h_b^i = f(x_i, h_b^{i+1})$

What does Bi-LSTM Layer compute?



Revisiting the Deep Learning Architecture for NER

What does Bi-LSTM Layer compute?



Revisiting the Deep Learning Architecture for NER

Bi-LSTM Layer

- The Bi-LSTM layer extracts a set of features for every word in the sentence
- We will now call this representation as **instance-level representation**
- Consider the sentence snippets,
  - वर्तमान में **उत्तर** प्रदेश के जौनसार बावर क्षेत्र …
    - Currently Jaunsar Bavar area of Uttar Pradesh ...
  - … भावजूद भी कोई प्रोत्साहित (संतोषजनक) **उत्तर** प्राप्त नहीं हुआ
    - even after that no satisfactory answer was obtained
- The word **उत्तर** will now have two instance-level representations one for first sentence and the other for second sentence
- We will now query the nearest neighbors for उत्तर using instance-level representations from both sentences

B-LSTM Layer

Word Embedding			Sentence 1			Sentence 2		
Neighbors	**Score**	**Tag**	**Neighbors**	**Score**	**Tag**	**Neighbors**	**Score**	**Tag**
प्रदेश	0.8722	-	उत्तरी	0.9088	LOC	उत्तर	0.9183	O
पश्चिम	0.8596	-	उत्तर	0.9033	LOC	उत्तर	0.9155	O
मध्य	0.8502	-	तिब्बत	0.8669	LOC	उत्तर	0.9137	O
पूरब	0.8432	-	शिमला	0.8641	LOC	उत्तर	0.9125	O
अरुणाचल	0.8430	-	किन्नौर	0.8495	LOC	उत्तर	0.9124	O

- The table shows the nearest neighbors (using cosine similarity) for the ambiguous word उत्तर using instance-level representation
- In sentence 1, the nearest neighbors are all location entities
- In sentence 2, we observe different instances of उत्तर appearing as nearest neighbors
    - All the instances of उत्तर takes the answer meaning as in *… कि उत्तर देने वाले व्यक्ति ..*

Analyzing Bi-LSTM Layer

Sentence 2			
Neighbors	**Score**	**Tag**	**Sentence**
उत्तर	0.9183	O	कि उत्तर देने वाले व्यक्ति
उत्तर	0.9155	O	अनुसार उत्तर दिये परन्तु
उत्तर	0.9137	O	उसे उत्तर देने में
उत्तर	0.9125	O	सही उत्तर की संभावना
उत्तर	0.9124	O	एक भी उत्तर ना दे

- In sentence 2, we observe different instances of उत्तर appearing as nearest neighbors
  - All the instances of उत्तर takes the answer meaning as in *… कि उत्तर देने वाले व्यक्ति ..*

Softmax Layer (Linear + Softmax)

- The output from Bi-LSTM module and correct previous tag is fed as input to Softmax layer
- The correct previous tag is crucial in identifying the named entity phrase boundaries
- During testing, we do not have previous tag information
- We use beam search to find the best possible tag sequence

Results

- We perform the NER experiments on the following set of languages

Language	Dataset
English	CoNLL 2003 Shared Task
Spanish	CoNLL 2002 Shared Task
Dutch	CoNLL 2002 Shared Task
Hindi	IJCNLP 2008 Shared Task
Bengali	
Telugu	
Marathi	In-House Data

Results

- The following Table shows the F1-Score obtained using the Deep Learning system

Language	F1-Score
English	90.94
Spanish	84.85
Dutch	85.20
Hindi	59.80
Marathi	61.78
Bengali	43.24
Telugu	21.11

Demo

Thank You

Questions?

References

- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. J. Mach. Learn. Res.,
- dos Santos, C., Guimaraes, V., Niterói, R., and de Janeiro, R. (2015). Boosting named entity recognition with neural character embeddings. Proceedings of NEWS 2015 The Fifth Named Entities Workshop, page 9.
- Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. CoRR, abs/1508.01991
- Lample, G., Ballesteros, M., Kawakami, K., Subramanian, S., and Dyer, C. (2016). Neural architectures for named entity recognition. In In proceedings of NAACL-HLT (NAACL 2016)., San Diego, US

References

- Gillick, Dan and Brunk, Cliff and Vinyals, Oriol and Subramanya, Amarnag "Multilingual Language Processing From Bytes." *In proceedings of NAACL-HLT (NAACL 2016)., San Diego, US.*
- Murthy, Rudra and Bhattacharyya, Pushpak “Complete Deep Learning Solution For Named Entity Recognition.” *CICLING 2016, Konya, Turkey*
- Murthy, Rudra and Khapra, Mitesh and Bhattacharyya, Pushpak “Sharing Network Parameters for Crosslingual Named Entity Recognition” *CoRR, abs/1607.00198*