A Survey on Deep Learning for Named Entity Recognition

Jing Li et al.
Presentation by Maxim Tikhonov

May 14, 2024

Outline

1. Task definition
2. Evaluation metrics
3. Traditional approaches
4. Deep learning approach
5. Context encoder architectures
6. Tag decoder architectures
7. Applied Deep Learning

Task definition

$< w_1, w_3, \text{Person} >$ Michael Jeffrey Jordan

$< w_7, w_7, \text{Location} >$ Brooklyn

$< w_9, w_{10}, \text{Location} >$ New York

$\Uparrow$ $< I_s, I_e, t >$

Named Entity Recognition

$\Uparrow$ $s =< w_1, w_2, ..., w_N >$

Michael	Jeffrey	Jordan	was	born	in	Brooklyn	,	New	York	.
w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}

Evaluation metrics

- Exact-match evaluation

  consider a prediction correct if it has both boundaries and type matching ground truth.

- Relaxed-match evaluation

  credit a predicted type if it matches the ground truth and overlaps with ground truth boundaries; credit predicted boundaries if they match the ground truth boundaries regardless of a predicted type.

F-score

$$\text{F-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},$$

where $\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}$

Macro-averaged F-score treats all entity types equally (average is taken of scores for each entity type).

Micro-averaged F-score treats all entities equally (average is taken of scores for all entities).

Relaxed-match evaluation

Evaluation doesn't require the prediction to match both the boundary and entity type:

- Correct type is credited when the predicted type matches the ground truth and there is an overlap with ground truth boundaries.
- Correct boundaries are credited only when they match perfectly with ground truth.

Improvement (?): ACE proposed a metric that takes into account subtypes of named entities.

Rule-based approaches

- Reliance on hand-crafted/inferred rules.
- Regexp.
- Works well when we know enough language-specific information and resources (dictionaries, lexicons, linguists).
- Rules can't easily represent some dependencies.

Unsupervised learning approaches

- Clustering – gathering of named entities from clustered groups based on the similarity of context.
- Usage of hyperonyms/hyponyms (location>country, creature>animal>bear).
- Querying the web/database for patterns (Google queries like "such as X").
- Mining named entities from several newspapers at time X.
- Reliance on lexical resources (e.g. word net), lexical patterns.

Feature-based approaches

- Feature engineering.

- Features – descriptors or characteristic attributes of words designed for algorithmic consumption (abstraction layer over the text/words).

Word-level features

Case : capitalization, uppercase, mixed case.

Punctuation : word has a period (ends with it, letters are separated by it)/apostrophe/hyphen/ampersand.

Digit : digit patterns (dates, time, IDs, serial numbers, ...), Roman numerals, abbreviations with digits.

Morphology : utilization of morphemes: {pre-,suf-}fixes, root words.

Part-of-speech : proper name, verb, noun, foreign word.

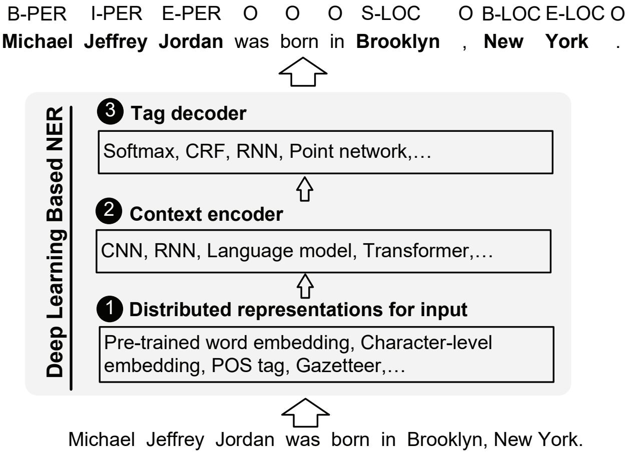
List lookup features

General list : stop words, capitalized nouns (months/days of the week), abbreviations.

List of entities : organizations, names, geographical entities

List of entity cues : name prefixes, titles, typical words in organization names.

Deep Learning techniques



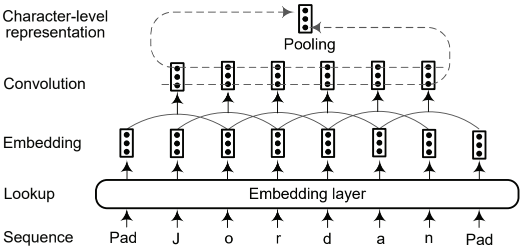
Input representation

Word-level

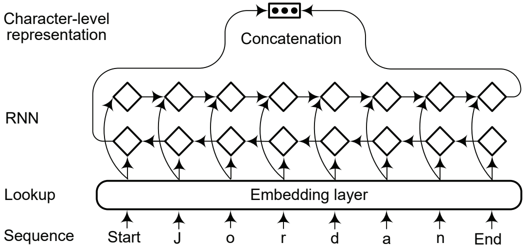
- One-hot, Word2vec (CBOW/Skip-gram), Glove, fastText, Bert, ...
- Use pre-trained word embeddings.
- One of the mentioned works utilizes a model that is trained for two sub-tasks: it first segments the text, and then predicts labels.

Input representation

Character-level



CNN-based char-level representation



RNN-based char-level representation

Input representation

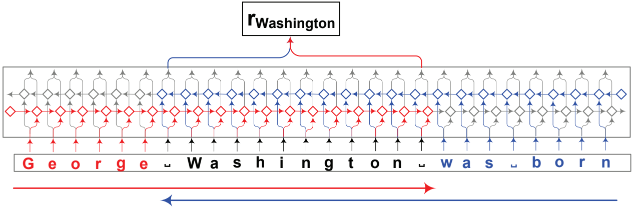
Methodologies

- Use representations from both levels – first, extract char-level word representations using CNN and concatenate them with word embedding (can also add a gate to make the model decide which representation to utilize more), then feed it into a (bidirectional) RNN context encoder.

- Consider a sentence as a sequence of characters and apply utilize RNN (LSTM) to extract char-level representation. Output a tag distribution for each character.

Input representation

- Contextual string embeddings: use string char-level LM to generate contextualized embeddings for a string in a sentence context.



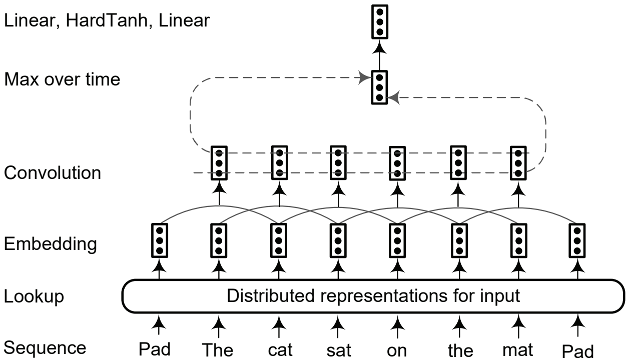
Forward (red) LM extract the hidden state after the last character in the word, backward (blue) LM extracts the hidden state before the first one.

Input representation

Hybrid – a combination of feature-based and neural approaches

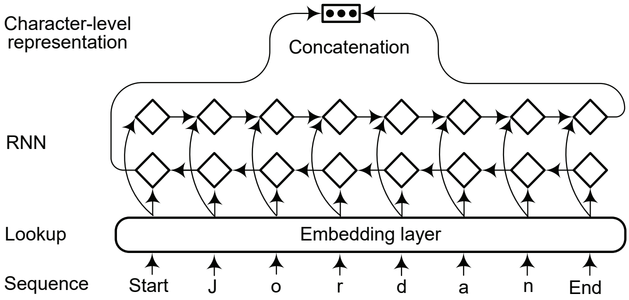
- Yields even better results than neural approaches.
- Systems employ rich features such as POS tags, morphological features, capitalization, etc.
- Resulting representations are often concatenations of embeddings vector and vector of features.

CNNs



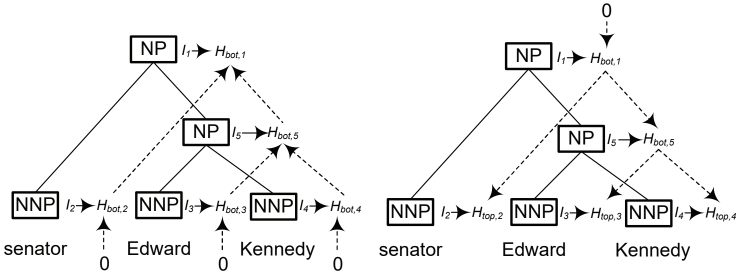
Each word is embedded to a multidimensional vector, then a convolution layer is applied to produce features around each word, then a global feature vector is built by combining these features. Both local and global features are then fed to a linear NN for decoding.

RNNs



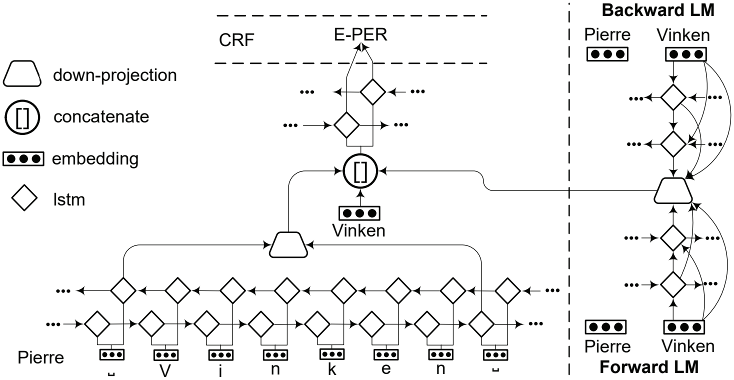
BRNN allows the model to contain information from the whole input sequence.

Recursive NNs



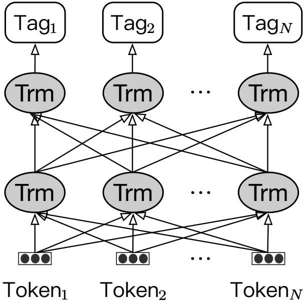
Recursive Neural Nets allow us to parse the sentence node by node using a constituency structure. The bottom-up direction calculates the semantic composition of the subtree of each node, and the top-down counterpart propagates to that node the linguistic structures which contain the subtree.

Neural LLMs

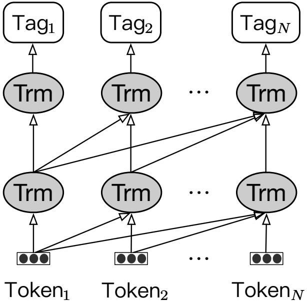


LM-LSTM-CRF model. The language model and sequence tagging model share the same character-level layer in a multi-task learning manner.

Transformer



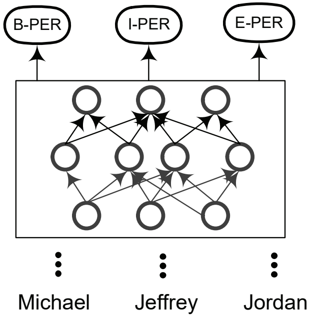
BERT



GPT

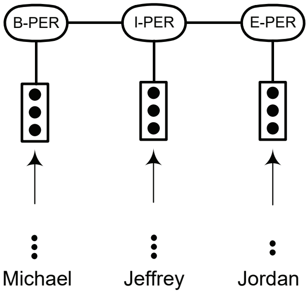
Just as in previous architectures, transformer embeddings are contextualized.

MLP + softmax

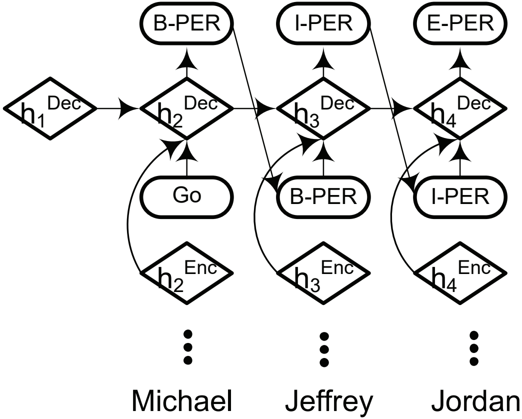


MLP+softmax decoder reformulates the NER problem to a multi-class classification problem. Tag for each word is independently predicted based on context-dependent representations.

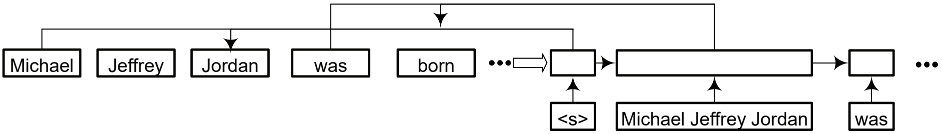
Conditional Random Fields



RNNs


B-PER
I-PER
E-PER
h1 Dec
h2 Dec
h3 Dec
h4 Dec
Go
B-PER
I-PER
h2 Enc
h3 Enc
h4 Enc
Michael
Jeffrey
Jordan

Pointer networks



Pointer networks first identify a segment and then label it.

Multi-task Learning

Let the model discover internal representations that are useful for many tasks.
Approaches:

- Train a model to jointly perform additional tasks like POS tagging, segmentation, SRL (Semantic Role Labeling).
- Modelling NER as two related subtasks: entity segmentation and category prediction.

Deep Learning Transfer

Approaches:

- Bootstrapping

Deep Reinforcement Learning

"Maximizing some heuristic helps to train a better performing model: the agent learns from the environment by interacting with it and receiving rewards."

Key components of the environment

- State transition function.
- Output function.
- Reward function

Key components of the agent:

- State transition function.
- Policy function.

Deep Adversarial Learning

"Learn to generate from a training distribution through a 2-player game: one network generates candidates, and the other evaluates them"
Adversarial examples can be produced in 2 ways

1. Consider instances in a source domain as adversarial examples for a target domain, and vice versa.
2. Prepare an adversarial sample by adding an original sample with a perturbation. Useful when dealing with a low-resource setting. The classifier is trained on both original and adversarial examples.

Neural attention

Application:

- When combining word-level and char-level input representations, use the attention mechanism to make the model decide what representations are more important.
- Obtaining relevant information from the entire document.