Named Entity Recognition from a Data-Driven Perspective

Jingbo Shang
Computer Science Engineering & Halıcıoğlu Data Science Institute
University of California, San Diego
Background

What’s Named Entity Recognition (NER)?

What’s “Data-Driven”?  

Data-Driven NER Methods
What’s Named Entity Recognition?

Wikipedia:

- **Named-entity recognition (NER)** is a subtask of information extraction (IE) that seeks to locate and classify named entities in text into pre-defined categories.
- In IE, a named entity is a real-world object.

Example

- Input
  - Jim bought 300 shares of Acme Corp. in 2006.

- Output
Supervised Methods: Training Data

- Sequence labeling framework
- Two popular schemes
  - BIO: **Begin**, **In**, **Out**
  - BIOES: **Begin**, **In**, **Out**, **End**, **Singleton**
  - BIOES is arguably better than BIO (Ratinov and Roth, ACL 09)
- Example:
  - **LABELS:** 
    
    | Jim | Person | bought | 300 | shares | of | Acme | Corp. | Organization | in | 2006 | Time |
    |-----|--------|--------|-----|--------|----|------|-------|--------------|----|------|------|
  - **TOKNES:** 
    
    | Jim | bought | 300 | shares | of | Acme | Corp. | in | 2006 | . |
  - **BIO:** 
    
    | B-PER | 0 | 0 | 0 | 0 | B-ORG | I-ORG | 0 | B-Time | 0 |
  - **BIOES:** 
    
    | S-PER | 0 | 0 | 0 | 0 | B-ORG | E-ORG | 0 | S-Time | 0 |
Supervised Methods: Neural Models

- Two pioneer models
  - LSTM-CRF (Lample et al., NAACL’16)
  - LSTM-CNN-CRF (Ma and Hovy, ACL’16)

<table>
<thead>
<tr>
<th></th>
<th>LSTM-CRF</th>
<th>LSTM-CNN-CRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word-Level</td>
<td>Bidirectional LSTMs</td>
<td>Bidirectional LSTMs</td>
</tr>
<tr>
<td>Character-Level</td>
<td>Bidirectional LSTMs</td>
<td>Convolutional NN</td>
</tr>
</tbody>
</table>

- The first neural model that outperforms the models based on handcrafted features
“Data-Driven” Philosophy

Key
- Enhance NER performance without introducing any additional human annotations

Questions
- Can massive raw texts help?
- Can dictionaries help?
- Are human annotations always correct?
- Is Tokenizer always good?
- ...
Questions

- Can massive raw texts help?
- Can dictionaries help?
- Are human annotations always correct?
- Is Tokenizer always good?
Using **Language Model** for better representations:

- **Word-level** Language Model:
  - ELMo (Peters et al., NAACL’18, best paper)
  - LD-Net (Liu et al., EMNLP’18)

- **Char-level** Language Model:
  - LM-LSTM-CRF (Liu et al., AAAI’18)
  - Flair (Akbik et al., COLING’18)

- **Hybrid** Language Model:
  - Cross View Training (Clark et al., EMNLP’ 2018)
  - BERT (Devlin et al., NAACL’19, best paper)
What’s (Neural) Language Model?

- Describing the generation of text:
  - predicting the next word based on previous contexts

- Pros:
  - Does not require any human annotations
  - Nearly **unlimited training data**!
  - Resulting models can generate sentences of an unexpectedly high quality
Char-by-Char Markdown Generations:

""See also"": [[List of ethical consent processing]]

== See also ==
*[[Iender dome of the ED]]
*[[Anti-autism]]

===[[Religion|Religion]]===
*[[French Writings]]
*[[Maria]]
*[[Revelation]]
Deep “Donald Trump”: Mimic President Trump

We have competence. Our people don’t need anybody. I have smart people.

I'm a Neural Network trained on Trump's transcripts. Priming text in [ ]. Donate (gofundme.com/deepdrumpf) to interact! Created by @hayesbh.

Followed by 7 users, 26K followers.
Propose to use **Character-level language model** as a **Co-Training** objective

Why character-level?

- More efficient & More robust to pre-processing
ELMo: Pre-train Word-Level Neural LM

- Add ELMo at the input of RNN. For some tasks (SNLI, SQuAD), including ELMo at the output brings further improvements

- Key points:
  - **Freeze** the weight of the biLM
  - Regularization are necessary
Make the contextualized represent **efficient without** much loss of effectiveness.

<table>
<thead>
<tr>
<th>Network</th>
<th>Avg. ppl</th>
<th>#FLOPs $\cdot 10^6$</th>
<th>$F_1$ score (avg±std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoLM (/)</td>
<td>/</td>
<td>3</td>
<td>90.78±0.24</td>
</tr>
<tr>
<td>O-ELMo (3)</td>
<td>39.70</td>
<td>607†</td>
<td>92.22±0.10</td>
</tr>
<tr>
<td>R-ELMo (6)</td>
<td>40.27</td>
<td>215</td>
<td>91.99±0.24</td>
</tr>
<tr>
<td>R-ELMo (7)</td>
<td>48.85</td>
<td>135</td>
<td>91.54±0.10</td>
</tr>
<tr>
<td>TagLM (5)</td>
<td>47.50</td>
<td>87†</td>
<td>91.62±0.23</td>
</tr>
<tr>
<td>LD-Net (8)</td>
<td>45.14</td>
<td>98</td>
<td>91.76±0.18</td>
</tr>
<tr>
<td>LD-Net (9)</td>
<td>50.06</td>
<td>98</td>
<td>91.86±0.15</td>
</tr>
<tr>
<td>LD-Net (8*)</td>
<td>origin</td>
<td>98</td>
<td>91.95</td>
</tr>
<tr>
<td></td>
<td>pruned</td>
<td>6</td>
<td>91.55±0.06</td>
</tr>
<tr>
<td>LD-Net (9*)</td>
<td>origin</td>
<td>98</td>
<td>92.03</td>
</tr>
<tr>
<td></td>
<td>pruned</td>
<td>6</td>
<td>91.84±0.14</td>
</tr>
</tbody>
</table>
Flair: Pre-Train Neural LM at All Levels

- Even for character-level language model, pre-training is very important.
- The structure is the same with LM-LSTM-CRF, the difference is the pre-training conducted on additional training corpus.

<table>
<thead>
<tr>
<th>Task</th>
<th>PROPOSED</th>
<th>Previous best</th>
</tr>
</thead>
<tbody>
<tr>
<td>NER English</td>
<td>93.09±0.12</td>
<td>92.22±0.1 (Peters et al., 2018)</td>
</tr>
<tr>
<td>NER German</td>
<td>88.32±0.2</td>
<td>78.76 (Lample et al., 2016)</td>
</tr>
<tr>
<td>Chunking</td>
<td>96.72±0.05</td>
<td>96.37±0.05 (Peters et al., 2017)</td>
</tr>
<tr>
<td>PoS tagging</td>
<td>97.85±0.01</td>
<td>97.64 (Choi, 2016)</td>
</tr>
</tbody>
</table>
BERT: Introduce Transformer

- Introduce Transformers, use masked language model + next sentence prediction
- Conduct fine-tuning after pre-training on each task (necessary for sentence-level tasks, NER is a word level task).

<table>
<thead>
<tr>
<th>System</th>
<th>Dev F1</th>
<th>Test F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELMo (Peters et al., 2018a)</td>
<td>95.7</td>
<td>92.2</td>
</tr>
<tr>
<td>CVT (Clark et al., 2018)</td>
<td>-</td>
<td>92.6</td>
</tr>
<tr>
<td>CSE (Akbik et al., 2018)</td>
<td>-</td>
<td>93.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fine-tuning approach</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT\textsubscript{LARGE}</td>
<td>96.6</td>
<td>92.8</td>
</tr>
<tr>
<td>BERT\textsubscript{BASE}</td>
<td>96.4</td>
<td>92.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature-based approach (BERT\textsubscript{BASE})</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Embeddings</td>
<td>91.0</td>
<td>-</td>
</tr>
<tr>
<td>Second-to-Last Hidden</td>
<td>95.6</td>
<td>-</td>
</tr>
<tr>
<td>Last Hidden</td>
<td>94.9</td>
<td>-</td>
</tr>
<tr>
<td>Weighted Sum Last Four Hidden</td>
<td>95.9</td>
<td>-</td>
</tr>
<tr>
<td>Concat Last Four Hidden</td>
<td>96.1</td>
<td>-</td>
</tr>
<tr>
<td>Weighted Sum All 12 Layers</td>
<td>95.5</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 7: CoNLL-2003 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.
New State-of-the-arts

- Using **Language Model** for better representations:
  - Word-level Language Model:
    - ELMo (Peters et al., NAACL’18, **best paper**) 92.2
    - LD-Net (Liu et al., EMNLP’18) 92.0, ~5X faster
  - Char-level Language Model:
    - LM-LSTM-CRF (Liu et al., AAAI’ 18) 91.4
    - Flair (Akbik et al., COLING’18) **93.1**
  - Hybrid Language Model:
    - Cross View Training (Clark et al., EMNLP’ 2018) 92.6
    - BERT (Devlin et al., NAACL’19, **best paper**) 92.4 / 92.8
Questions

- Can massive raw texts help? → Neural language model
- Can dictionaries help?
- Are human annotations always correct?
- Is Tokenizer always good?
Distantly Supervised NER

- **Input**
  - Unlabeled Raw Texts
  - An Entity Dictionary
    - entity type, canonical name, [synonyms_1, synonyms_2, ..., synonyms_k]

- **Output**
  - A NER model to recognize the entities of the entity types appeared in the given dictionary.
  - Note that the entities to be recognized can be unseen entities.
Distantly Supervised NER Methods

- String-match / rule-based distant supervision generation

- AutoEntity, SwellShark, ClusType, ...
  - Leave the entity span detection to experts
  - POS Tag Rule-based (e.g., regular expressions)

- Distant-LSTM-CRF
  - Leverage AutoPhrase to extract “aspect terms”

- AutoNER
  - A novel “Tie-or-Break” labeling scheme + tailored neural model
SwellShark: Distantly Supervised Typing

- Data Programming for Typing
- Entity Span Detection: Regular expressions based on part-of-speech (POS) tags
  - Requires expert efforts
  - Candidate Generators
Distant-LSTM-CRF: Use Phrase Mining as Supervision + LSTM-CRF

- AutoPhrase + LSTM-CRF
  - AutoPhrase generates labels
  - Heuristically set thresholds
  - LSTM-CRF builds models
  - Both word & char info are used

Problem
- High thresholds needed for clean positive labels
  - many false-negative labels
AutoNER: Dual Dictionaries

- A core dictionary
  - Leads to high-precision but low-recall matches
- A “full” dictionary
  - Leads to high-recall but low-precision matches
  - Introduce out-of-dictionary high-quality phrases as new entities
    - Their types are “unknown”
    - It could be any IOBES + any type
Figure 1: The illustration of the Fuzzy CRF layer with modified IOBES tagging scheme. The named entity types are \{Chemical, Disease\}. “indomethacin” is a matched Chemical entity and “prostaglandin synthesis” is an unknown-typed high-quality phrase. Paths from Start to End marked as purple form all possible label sequences given the distant supervision.
Instead of labeling each token, we choose to tag the connection between two adjacent tokens.

For every two adjacent tokens, the connection between them is labeled as

1. **Tie**, when the two tokens are matched to the same entity
2. **Unknown**, if at least one of the tokens belongs to an unknown-typed high-quality phrase;
3. **Break**, otherwise.
AutoNER: Tailored Neural Model

Figure 2: The illustration of AutoNER with Tie or Break tagging scheme. The named entity type is \{AspectTerm\}. “ceramic unibody” is a matched AspectTerm entity and “8GB RAM” is an unknown-typed high-quality phrase. Unknown labels will be skipped during the model training.
Comparison – Biomedical Domain

Table 2: [Biomedical Domain] NER Performance Comparison. The supervised benchmarks on the BC5CDR and NCBI-Disease datasets are LM-LSTM-CRF and LSTM-CRF respectively (Wang et al., 2018). SwellShark has no annotated data, but for entity span extraction, it requires pre-trained POS taggers and extra human efforts of designing POS tag-based regular expressions and/or hand-tuning for special cases.

<table>
<thead>
<tr>
<th>Method</th>
<th>Human Effort other than Dictionary</th>
<th>BC5CDR</th>
<th>NCBI-Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Pre</td>
<td>Rec</td>
</tr>
<tr>
<td>Supervised Benchmark</td>
<td>Gold Annotations</td>
<td>88.84</td>
<td>85.16</td>
</tr>
<tr>
<td>SwellShark</td>
<td>Regex Design + Special Case Tuning</td>
<td>86.11</td>
<td>82.39</td>
</tr>
<tr>
<td></td>
<td>Regex Design</td>
<td>84.98</td>
<td>83.49</td>
</tr>
<tr>
<td>Dictionary Match</td>
<td>None</td>
<td>93.93</td>
<td>58.35</td>
</tr>
<tr>
<td>Fuzzy-LSTM-CRF</td>
<td>None</td>
<td>88.27</td>
<td>76.75</td>
</tr>
<tr>
<td>AutoNER</td>
<td></td>
<td>88.96</td>
<td>81.00</td>
</tr>
</tbody>
</table>
Questions

- Can massive raw texts help? → Neural language model
- Can dictionaries help? → Distant supervised setting
- Are human annotations always correct?
- Is Tokenizer always good?
Typical Annotation Mistakes in CoNLL03

- State-of-the-art F1 score on this test set is already around 93%
- ~5.38% test sentences have annotation mistakes
  - Significant amount!

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Original labels</th>
<th>Corrected labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sporting Gijon 15 4 4 7 15 22 16</td>
<td>[Sporting]{ORG}</td>
<td>[Sporting Gijon]{ORG}</td>
</tr>
<tr>
<td>SOCCER - JAPAN GET LUCKY WIN,</td>
<td>[JAPAN] {LOC}, [China]{PER}</td>
<td>[JAPAN] {LOC}, [China]{LOC}</td>
</tr>
<tr>
<td>CHINA IN SURPRISE DEFEAT.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NZ First on Sunday.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Seagramd ace 20/11/96 5,000 Japan</td>
<td>[Seagramd] {MISC}, [Japan]{LOC}</td>
<td>[Seagramd ace] {MISC}, [Japan] {LOC}</td>
</tr>
</tbody>
</table>
Evaluation on Corrected Test Set

- Higher F1 score with smaller variance
- Better reflects the real performance
- This corrected test set should be adopted in future research

<table>
<thead>
<tr>
<th>Method</th>
<th>Original</th>
<th>Corrected</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM-CRF</td>
<td>90.64 (±0.23)</td>
<td>91.47 (±0.15)</td>
</tr>
<tr>
<td>LSTM-CNNs-CRF</td>
<td>90.65 (±0.57)</td>
<td>91.87 (±0.50)</td>
</tr>
<tr>
<td>VanillaNER</td>
<td>91.44 (±0.16)</td>
<td>92.32 (±0.16)</td>
</tr>
<tr>
<td>Elmo</td>
<td>92.28 (±0.19)</td>
<td>93.42 (±0.15)</td>
</tr>
<tr>
<td>Flair</td>
<td>92.87 (±0.08)</td>
<td>93.89 (±0.06)</td>
</tr>
<tr>
<td>Pooled Flair</td>
<td>93.14 (±0.14)</td>
<td>94.13 (±0.11)</td>
</tr>
</tbody>
</table>
CrossWeigh: Handle Noisy Training Set

- Original Training Set:
  - [Liverpool]{ORG} 3:2 ...
  - ... live in [Chicago]{LOC}.
  - [Chicago]{LOC} won ...

- Previous NER Model:
  - [Lakers]{LOC} won ...

- Many mistakes similar to “[Chicago]{LOC} won ...” make the NER model learn a wrong “LOC won” pattern.

- Our framework automatically identifies such mistakes and lowers their weights in training.

- Partition into k folds:
  - [Liverpool]{ORG} 3:2 ...
  - ...
  - ... live in [Chicago]{LOC}.
  - [Chicago]{LOC} won ...

- Training Set for k-th fold:
  - [Liverpool]{ORG} 3:2 ...
  - ...
  - ... live in [Chicago]{LOC}.
  - [Chicago]{LOC} won ...

- Identify Potential Mistakes:
  - ✓ ... live in [Chicago]{LOC}.
  - ? [Chicago]{LOC} won ...
  - ... live in [Chicago]{LOC}.
  - [Chicago]{ORG} won ...

- Weighted Training Set:
  - 1.0 [Liverpool]{ORG} 3:2 ...
  - ...
  - ... live in [Chicago]{LOC}.
  - 0.9 [Chicago]{LOC} won ...
  - 0.1 [Chicago]{LOC} won ...

- NER Model for k-th fold:

- NER Model Trained with Our Framework:
  - ✓ [Lakers]{ORG} won ...
Key Problem: How to Partition k-Folds?

- Random Partition may be ineffective
  - Neural NER models will overfit the annotation mistakes observed during training

- Entity Disjoint Filtering
  - In each fold, if a “training” sentence contains any entities appeared in the “testing” set, it will be discarded during the “training”
CrossWeigh: Evaluation

- CrossWeigh is effective with many NER models
  
<table>
<thead>
<tr>
<th></th>
<th>Original CoNLL03</th>
<th>Corrected CoNLL03</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>w/o CrossWeigh</td>
<td>w/ CrossWeigh</td>
</tr>
<tr>
<td>VanillaNER</td>
<td>91.44 (±0.16)</td>
<td>91.78 (±0.06)</td>
</tr>
<tr>
<td>Flair</td>
<td>92.87 (±0.08)</td>
<td>93.19 (±0.09)</td>
</tr>
<tr>
<td>Pooled-Flair</td>
<td>93.14 (±0.14)</td>
<td>93.43 (±0.06)</td>
</tr>
</tbody>
</table>

- Entity Disjoint Filtering is important

- Twitter & Low-resource

<table>
<thead>
<tr>
<th>Method</th>
<th>Original</th>
<th>Corrected</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o CrossWeigh</td>
<td>92.87 (±0.08)</td>
<td>93.89 (±0.06)</td>
</tr>
<tr>
<td>w/ CrossWeigh</td>
<td>93.19 (±0.09)</td>
<td>94.18 (±0.06)</td>
</tr>
<tr>
<td>- Entity Disjoint</td>
<td>92.88 (±0.11)</td>
<td>93.84 (±0.08)</td>
</tr>
<tr>
<td>+ Random Discard</td>
<td>93.01 (±0.10)</td>
<td>93.94 (±0.10)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>w/o CrossWeigh</th>
<th>w/ CrossWeigh</th>
</tr>
</thead>
<tbody>
<tr>
<td>WNUT’17</td>
<td>48.96 (±0.97)</td>
<td>50.03 (±0.40)</td>
</tr>
<tr>
<td>Sinhalese</td>
<td>66.34 (±0.34)</td>
<td>67.68 (±0.21)</td>
</tr>
</tbody>
</table>
CrossWeigh: Identify Annotation Mistakes

- CoNLL03 train, dev & test as a super training set
- Apply CrossWeigh to identify annotation mistakes on the test set
- Evaluate against 186 human corrections
  - Almost 80% of mistakes can be detected

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Potential Mistakes</th>
<th>Actual Mistakes</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flair</td>
<td>573.0</td>
<td>144.0</td>
<td>0.2513</td>
<td>0.7742</td>
<td>0.3794</td>
</tr>
<tr>
<td>VanillaNER</td>
<td>821.67</td>
<td>146.33</td>
<td>0.1781</td>
<td>0.7867</td>
<td>0.2904</td>
</tr>
</tbody>
</table>
Questions

- Can massive raw texts help? → Neural language model
- Can dictionaries help? → Distant supervised setting
- Are human annotations always correct? → Auto-Correction
- Is Tokenizer always good?
Typical NER Pipeline System

- Pre-processing tools are applied first
An Interesting Observation

- Broad Twitter Corpus (BTC)
  - A twitter NER dataset

- spaCy
  - A popular Python NLP lib

- spaCy tokenization + BTC dataset
  - Word boundaries of more than 45% named entities will be incorrectly identified!
We propose to conduct NER training in a raw-to-end manner

Raw text as the input & Predictions at the character level
Neural-Char-CRF: String Match

- Prefer to match the words with higher Inverse Document Frequency (IDF)
Neural-Char-CRF: Character-Level LM

- Character-level neural language model is leveraged
- Pre-training + Contextualized representations

![Diagram showing language model pre-training and integrated model](image-url)
### Comparison – Twitter NER Datasets

- **Tokenizer matters**
- **NLTK is the best on both datasets**
- **Raw-to-End wins**
- **String Match is even better**

<table>
<thead>
<tr>
<th>Tokenizer</th>
<th>Methods</th>
<th>TNT</th>
<th>BTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLTK</td>
<td>TwitterNER</td>
<td>73.41</td>
<td>64.40</td>
</tr>
<tr>
<td></td>
<td>LSTM-CNN-CRF</td>
<td>80.25</td>
<td>66.48</td>
</tr>
<tr>
<td></td>
<td>LM-LSTM-CRF</td>
<td>80.85</td>
<td>67.73</td>
</tr>
<tr>
<td></td>
<td>Flair</td>
<td><strong>83.26</strong></td>
<td><strong>68.33</strong></td>
</tr>
<tr>
<td>spaCy</td>
<td>TwitterNER</td>
<td>70.40</td>
<td>37.21</td>
</tr>
<tr>
<td></td>
<td>LSTM-CNN-CRF</td>
<td>71.66</td>
<td>29.04</td>
</tr>
<tr>
<td></td>
<td>LM-LSTM-CRF</td>
<td>72.25</td>
<td>31.17</td>
</tr>
<tr>
<td></td>
<td>Flair</td>
<td>73.49</td>
<td>33.20</td>
</tr>
<tr>
<td>Stanford</td>
<td>TwitterNER</td>
<td>70.18</td>
<td>38.01</td>
</tr>
<tr>
<td></td>
<td>LSTM-CNN-CRF</td>
<td>64.22</td>
<td>29.71</td>
</tr>
<tr>
<td>CoreNLP</td>
<td>LM-LSTM-CRF</td>
<td>63.14</td>
<td>28.97</td>
</tr>
<tr>
<td></td>
<td>Flair</td>
<td>65.58</td>
<td>32.39</td>
</tr>
<tr>
<td></td>
<td>Neural-Char-CRF (Match)</td>
<td><strong>86.96</strong></td>
<td><strong>74.98</strong></td>
</tr>
<tr>
<td></td>
<td>Neural-Char-CRF (NLTK)</td>
<td>85.59</td>
<td>74.39</td>
</tr>
</tbody>
</table>
- Using neural language model, massive raw texts can help!
- High-quality dictionaries can help!
- Human annotations are NOT always correct!
- Tokenizer is not that important and sometimes even hurts!