Holistic Evaluation of Named Entity Recognition Systems

Gabriel Bernier-Colborne

National Research Council Canada

4 October 2019

Outline

1 Background
2 Overview
3 Data
4 Systems
5 Methodology
6 Results: conlleval
7 Results: hard_eval
8 Concluding remarks

Named Entity Recognition

- Named Entity Recognition (NER):
  - Given a text, detect expressions that refer to entities, e.g. people, locations, organizations, etc.
  - Identify the type of entity
- NER is used in various applications
  - Information extraction, information retrieval, question-answer systems, etc.

NER Evaluation

- Standard evaluation method in research papers:
  - Automatic, quantitative evaluation on a benchmark, i.e. an annotated corpus of texts
    - Very often: CoNLL-2003 or OntoNotes
  - Systems trained and tested on data from the same distribution

Practical Issues

- Doing NER in practice raises questions that are often ignored in research papers
  - What accuracy will this or that approach achieve across various text types or domains?
  - What accuracy can I expect if I have very little training data for my target domain, or no data at all?
  - What can I do to maximize accuracy on a new domain?

Outline

1 Background
2 Overview
3 Data
4 Systems
5 Methodology
6 Results: conlleval
7 Results: hard_eval
8 Concluding remarks

Our Contributions

- Comparative evaluation of various NER systems
  - Both in-domain and across domains
- Code developed for these experiments
  - `https://github.com/gbcolborne/ner_eval`
- Observations on the limits of the standard evaluation paradigm
- Observations on what NER systems actually learn
- Tips on how to do NER in a new domain

Outline

1 Background
2 Overview
3 Data
4 Systems
5 Methodology
6 Results: conlleval
7 Results: hard_eval
8 Concluding remarks

Data

- 5 annotated corpora
- Various domains and text types
- Various challenging properties
  - Formal or informal language
  - # entity mentions in the training set (more makes learning easier)
  - # entity types (more makes type classification harder)
  - % unseen mentions in the test set (more makes detection harder)
  - % ambiguous mentions in the training set (entity mentions that belong to more than one entity type may be harder to classify)

Data

	CoNLL	WNUT	FIN	i2b2	ONTO
Domain	News	Twitter	Loans	Medical	Various
# entity types	4	6	4	23	18
# mentions (train)	23499	1975	460	11791	81828
# mentions (dev)	5942	836	0	5453	11066
# mentions (test)	5648	1079	120	11360	11257
% unseen mentions	46.0	100.0	70.8	81.2	32.0
% ambig. mentions	6.9	1.6	1.3	0.9	25.2

Outline

1 Background
2 Overview
3 Data
4 Systems
5 Methodology
6 Results: conlleval
7 Results: hard_eval
8 Concluding remarks

Background

- NER $\approx$ sequence labeling
- Sentence $\approx$ sequence of tokens
- Tokens are represented using features, which are based on the token and its context
- An NER system learns to predict a label for each token based on its features
- We use the IOB-2 (inside-outside-beginning) scheme to encode labels
  - E.g. $Virginia_{B\text{-}PER}\ Woolf_{I\text{-}PER}\ was_{O}\ born_{O}\ in_{O}\ London_{B\text{-}LOC}$

Background

- 2 ways to get features:
  1. Feature engineering (i.e. humans think of useful features and incrementally improve the feature set)
     - Lexical features (i.e. the token itself)
     - Sub-word features (e.g. case, affixes)
     - Contextual features (e.g. surrounding words)
     - Knowledge-based features (which exploit external resources, e.g. dictionaries or knowledge bases)
  2. Automatic feature extraction
     - Representation learning
     - Deep learning

Systems evaluated

1. Baseline: memorization (i.e. dictionary)
   - Can not handle unseen or ambiguous mentions
2. CRF++: generic CRF implementation for sequence labeling
   - I implemented a tiny feature set (token + surrounding words)
3. Stanford NER: CRF with rich feature set for NER
4. Illinois NER: Perceptron with even richer feature set
5. NeuroNER: neural network
   - Feature extractors (i.e. BiLSTM) at character and word level
   - CRF output layer
6. SpaCy: neural network
   - Feature extractor (i.e. CNN+Attention) at word level + injection of hand-crafted sub-word features
   - Stack-LSTM rather than CRF

Outline

1 Background
2 Overview
3 Data
4 Systems
5 Methodology
6 Results: conlleval
7 Results: hard_eval
8 Concluding remarks

Methodology

- Comparative evaluation of systems on various benchmarks
- 3 training methods:
  - In-domain (ID): use training set of same corpus as test set
  - Out-of-domain (OD): use all training sets EXCEPT the in-domain training set
  - All-domain (AD): use all training sets

Methodology

- Problem: what if the entity types annotated in the training data and test data are not the same?
- We map all entity types to a common set of 4 generic types (i.e. PER, ORG, LOC, MISC)
  - PATIENT $\rightarrow$ PER
  - CORPORATION $\rightarrow$ ORG
  - etc.
- Entity types that have no analogs in other corpora (e.g. numeric types in Ontonotes) are discarded

Methodology

- Automatic, quantitative evaluation
  - `conlleval` script
  - Compares predicted mentions to true mentions
  - Metrics: precision, recall, f-score ($\beta = 1$)
- Automatic error analysis
  - `error_analysis.py`
  - Enumerates false positives, false negatives, partial matches, misclassifications
- “Black box” evaluation
  - We use the default or recommended configuration of each system
  - When forced to select a hyperparameter or feature set, we do minimal optimization

Outline

1 Background
2 Overview
3 Data
4 Systems
5 Methodology
6 Results: conlleval
7 Results: hard_eval
8 Concluding remarks

Standard evaluation: in-domain training

- F-scores using ID training (no type mapping)
  - Note: results with type mapping are usually similar (1-2 points difference)

System	CoNLL	FIN	i2b2	ONTO	WNUT	Avg
Baseline	59.10	41.18	17.32	21.46	0.00	27.81
CRF++	69.59	28.57	61.61	76.10	0.19	47.21
Stanford	86.90	51.52	90.84	N/A	26.12	N/A
Illinois	**90.57**	**57.71**	90.80	83.53	29.53	70.43
NeuroNER	89.86	48.18	**92.68**	**86.47**	33.59	70.16
SpaCy	88.09	52.80	91.66	85.16	**35.16**	**70.57**

Standard evaluation: in-domain training

- Baseline (memorization):
  - 0% on WNUT $\rightarrow$ all unseen mentions
  - Almost 60 % on CoNLL
- FIN and WNUT are the hardest benchmarks
- On other benchmarks, scores are around 90%
  - So about 90% of predicted mentions are correct and about 90% of true mentions are correctly identified
- SpaCy produces the most robust results on average

Out-of-domain training

- F-scores using OD training (with type mapping)

System	CoNLL	FIN	i2b2	ONTO	WNUT	Avg
Baseline	25.20	2.46	1.40	44.97	4.43	15.69
CRF++	35.54	31.72	36.41	45.27	10.98	31.98
Stanford	65.00	14.52	22.21	67.57	30.55	39.97
Illinois	71.55	27.97	29.86	71.90	35.62	47.38
NeuroNER	**72.33**	**34.86**	**38.18**	**73.59**	**40.10**	**51.81**
SpaCy	68.27	15.34	22.67	70.59	36.54	42.68

Out-of-domain training

- Scores are much, much lower if we train out-of-domain
  - Especially on FIN, WNUT, and i2b2
  - Drop is lower on CoNLL and OntoNotes, because these 2 corpora cover similar domains (so less information loss)

All-domain training

- F-scores using AD training (with entity type mapping)

System	CoNLL	FIN	i2b2	ONTO	WNUT	Avg
Baseline	36.40	2.78	2.75	20.61	4.47	13.40
CRF++	66.79	47.17	67.22	76.45	12.04	53.93
Stanford	84.88	42.51	82.75	84.96	31.62	65.34
Illinois	88.36	35.93	81.19	86.12	36.24	65.57
NeuroNER	**89.78**	**51.94**	**88.93**	**88.82**	**39.86**	**71.87**
SpaCy	85.83	39.74	81.34	87.30	36.14	66.07

All-domain training

- Scores are much higher, but slightly lower than in-domain scores

Comparing the systems

- ID tests:
  - SpaCy is best on average
  - NeuroNER and Illinois are best on 2 tests each
- OD and AD tests:
  - NeuroNER is best on all tests
  - SpaCy is 3rd on OD test, performs poorly on FIN and i2b2
- So we would choose a different system if we use out-of-domain training data

Effect of adding in-domain data

- Let's compare the results of the OD and AD tests
- Q: what do we gain by adding in-domain data to the training set?

System	CoNLL	FIN	i2b2	ONTO	WNUT
Baseline	+11.20	+0.32	+1.35	-24.36	0.04
CRF++	+31.25	+15.45	+30.81	+31.18	+1.06
Stanford	+19.88	+27.99	+60.54	+17.39	+1.07
Illinois	+16.81	+7.96	+51.33	+14.22	+0.62
NeuroNER	+17.45	+17.08	+50.75	+15.23	-0.24
SpaCy	+17.56	+24.40	+58.67	+16.71	-0.40

Effect of adding in-domain data

- State-of-the-art systems typically gain 15-20 points when we add ID training data
  - Except on WNUT
- Gains are even greater (50+ points) on i2b2

Error analysis

- Distribution of errors made by NeuroNER on i2b2 test set

Training data	# false neg	# false pos	# partial match	# wrong type	Total
Out-of-domain	626	5185	867	218	6896
All-domain	364	175	133	59	731
In-domain	405	59	96	27	587

Error analysis

- So OD training $\rightarrow$ lots of false positives
- Examples of false positives:
  - Ailments: ED (PER), BRBPR (LOC), Alzheimer (PER), etc.
  - Procedures: Colonoscopy (PER), Lasix (MISC), Transcranial Doppler (PER), etc.
  - Medications: Zocor (ORG), NovoLog (PER), Proventil (LOC), etc.
  - Chemicals: O2 (LOC), Fe (LOC), Na (LOC), ETOH (PER), etc.
  - Measurements: BP (ORG), PTT (ORG), JVP (ORG), ABI (PER), NAD (LOC), etc.
  - Departments/disciplines: Cardiology (LOC), ED Observation Unit (ORG), Neuro / Psych (ORG), etc.

Error analysis

- Why does the model make these errors?
- Almost all these words are capitalized
  - The model relies too heavily on this cue
  - Adding ID data corrects this

Error analysis

- Ambiguity is also a source of errors
  - *Fe* is an *I-LOC* in the CoNLL training data, but it is *O* (not a mention) in i2b2
- In general, the definition of a given entity type may change from one domain to the next
  - It would tend to be more narrow in a given domain than in the set of all domains

Effect of adding out-of-domain data

- Let's compare the results of the ID and MD tests (with type mapping in both cases)
- Q: do we gain anything by adding OD data to our ID training data?

System	CoNLL	FIN	i2b2	ONTO	WNUT
Baseline	-22.70	-38.40	-11.85	-0.08	4.47
CRF++	-2.80	+18.60	+3.57	+0.91	+11.85
Stanford	-2.02	-9.01	-4.75	-0.78	+4.56
Illinois	-2.21	-21.78	-6.57	-1.09	+4.86
NeuroNER	-0.08	+3.76	-2.38	-0.38	+5.90
SpaCy	-2.26	-13.06	-9.49	-0.31	-0.50

Effect of adding out-of-domain data

- State-of-the-art systems typically lose a few points when we add OD data
  - Except on WNUT
  - Drop is greatest on FIN
- This suggests we should only use ID data (at least if we have enough of it)
- Note: there are smarter ways of using OD data (rather than just lumping it in with the ID data)
  - Sampling examples based on some criterion (e.g. density estimation)
  - Transfer learning

Outline

1 Background
2 Overview
3 Data
4 Systems
5 Methodology
6 Results: conlleval
7 Results: hard_eval
8 Concluding remarks

Evaluating hard cases

- The script `hard_eval.py` computes the **token error rate** (i.e. percentage of mislabeled tokens) on various **subsets** of tokens in the test set
  - **Unseen**: tokens that are not in the training set
    - Unseen-I: unseen tokens that are part of a mention
    - Unseen-O: unseen tokens that are not part of a mention
  - **Diff**: tokens whose label is not their most frequent label in the training set
    - Diff-I: token was usually O, but is I
    - Diff-O: token was usually I, but is O
    - Diff-E: token was usually I-X, but is I-Y (different entity type)

Evaluating hard cases

- The next slide shows the output of `hard_eval.py` on the predictions of NeuroNER on the CoNLL test set (using ID training).

Evaluating hard cases

Test tokens	Nb tokens	Nb words	Nb errors	Token error rate
all	46435	9488	1093	0.0235
unseen-I	2537	1494	306	0.1206
unseen-O	3119	2215	154	0.0494
unseen-all	5656	3693	460	0.0813
diff-I	201	70	77	0.3831
diff-O	215	91	108	0.5023
diff-etype	676	285	188	0.2781
diff-all	1092	431	373	0.3416
unseen+diff	6748	4124	833	0.1234

Evaluating hard cases

- Ambiguous words are even harder than unseen ones (higher error rate)
  - Same on OntoNotes
- Standard evaluation paradigm does not show this
  - Does not distinguish easy cases from hard ones
- Ambiguous and unseen tokens explain the vast majority of errors in this case (833/1093)
  - Even though they represent just a small fraction of the test tokens (6748/46435)

Evaluating hard cases

- We used simple heuristics to identify tokens that are likely to be hard
  - System-agnostic
  - Context-agnostic
- This allows for a fine-grained evaluation of NER systems
- Might also be used to train better systems
  - *Slice-based learning* (Chen et al., 2019)

Outline

1 Background
2 Overview
3 Data
4 Systems
5 Methodology
6 Results: conlleval
7 Results: hard_eval
8 Concluding remarks

On the standard evaluation paradigm

- Standard evaluation paradigm is not very informative
  - In-domain, limited number of domains/genres/languages
  - What results can we expect in the real world?
- Researchers strive to beat SOTA on CoNLL or OntoNotes, but hard to tell what their gains actually mean
  - Are they just better at modeling statistical noise, or even annotation errors?
  - Can also be hard to tell where gains come from, e.g. better model, better optimization or better data.

Revisiting evaluation

- Many recent works raise issues with standard evaluation methods in NLP:
  - *Ethical Considerations in NLP Shared Tasks* (Escartin et al., 2017)
  - *We need to talk about standard splits* (Gorman & Bedrick, 2019)
  - *AI competitions don't produce useful models* (Oakden-Rayner, 2019)
  - *How the Transformers broke NLP leaderboards* (Rogers, 2019)
  - *Are We Modeling the Task or the Annotator?* (Geva et al, 2019)
  - *Green AI* (Schwartz et al., 2019)

On our contribution

- Cross-domain evaluation of NER systems shows the importance of training on data similar to the data you expect to see in the real world
  - Huge drop in accuracy if we train out-of-domain
  - Even a small quantity of in-domain data can help a lot
- An evaluation focused on hard cases shows that ambiguity is even harder to handle than unseen words
- Simple heuristics can be used to identify the hard cases
- Might also be used to train better systems