Deep Learning Applications in Natural Language Processing

Jindřich Libovický

April 24, 2024
Today’s Learning Outcomes

After this lecture you should be able to ...

1. Tell how various NLP tasks can be formulated as a sequence labeling problem
2. Reason about problems with benchmarking in NLP (using Visual Question Answering and Answer Span Selection as an Example)
3. Describe how Named Entity Recognition and Answer Span Selection solved using pre-trained language representations
Named Entity Recognition

Answer Span Selection
Named Entity Recognition
Information Extraction = Subfield of NLP

Find **who** did **what** to **whom**.

Named entity recognition (NER) is one of the tasks.
The Mona Lisa is a 16th century oil painting created by Leonardo. It’s held at the Louvre in Paris.
NER: application areas

- Part of information extraction pipeline
  - Entity linking (e.g., matching Wikipedia articles)
  - Coreference resolution
    Whom does pronoun “they” refer to?
    Who is “the president” in a text?
- Indexing text for search
- Direct use in smart devices

NER used to create links in text to different apps.

Different entity recognizers use different sets. Example:

- **Person**: Names of people.
- **PersonType**: Job types or roles held by a person.
- **Location**: Natural and human-made landmarks, structures, geographical features, and geopolitical entities.
- **Organization**: Companies, political groups, musical bands, sport clubs, government bodies, and public organizations.
- **Event**: Historical, social, and naturally occurring events.
- **Product**: Physical objects of various categories.
- **Skill**: A capability, skill, or expertise.
- **Address**: Full mailing addresses.
- **Phone number**: Phone numbers.
- **Email**: Email addresses.
- **URL**: URLs to websites.
- **IP**: Network IP addresses.
- **DateTime**: Dates and times of day.
- **Quantity**: Numerical measurements and units.

List from Microsoft Text Analytics API (https://docs.microsoft.com/en-us/azure/cognitive-services/text-analytics/named-entity-types)
NER as Sequence Labeling

- Assign each token with a tag saying what entity it belongs to
- Different tagging schemes

**IOB Scheme**

I — Token is inside an entity.
O — Token is outside an entity.
B — Token is the beginning of an entity.

**BILUO Scheme** ← usually better

B — Token is the beginning of a multi-token entity.
I — Token is inside a multi-token entity.
L — Token is the last token of a multi-token entity.
U — Token is a single-token unit entity.
O — Token is outside an entity.


A sentence with 2 named entities:

<table>
<thead>
<tr>
<th>There</th>
<th>are</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quantity</th>
<th>compositions</th>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>over 1000</td>
<td></td>
<td>Johan</td>
</tr>
<tr>
<td>B-QUANT</td>
<td>I-QUANT</td>
<td>Sebastian</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>I-PERSON</td>
</tr>
</tbody>
</table>

Special B and I tags for each of the entity types.
Deep learning solution

As usual …

1. Embed input tokens as vectors
2. Contextualize the input embeddings using RNN/Transformer
3. Apply a classifier over each of the hidden states to predict the label

Current best solution:

Pre-trained Transformer (BERT) + finetuning for sequence labeling
Evaluation

**Precision**

\[ P = \frac{\text{# words correctly assigned to entities}}{\text{# words in all detected entities}} \]

Interpretation: How correct the system output is.

**Recall**

\[ R = \frac{\text{# words correctly assigned to entities}}{\text{# words in all ground-truth entities}} \]

Interpretation: How well the are the “real” entities covered.

**F-Score**

Harmonic mean of the previous two:

\[ F_1 = \frac{2PR}{P + R} \]

Reasonable numbers are >90% on standard datasets.
Adding Conditional Random Field

- Standard tagging: conditional independence assumption
  i.e., given the hidden states all predictions are independent
- Tags have their internal grammar
  e.g., I-PERSON can only follow I-PERSON or B-PERSON
- If the model is unsure about the tag, it can lead to inconsistent predictions

⇒ **Conditional Random Fields** (CRF) can help restrict the models to produce more consistent outputs.


Formal definition of the CRF

\[ P(y|h) = \frac{1}{Z} \exp \left( \sum_{i=1}^{n} \psi(y_i|h_i) + \sum_{i=1}^{n-1} \phi(y_i, y_{i+1}) \right) \]

\( y = (y_1, \ldots, y_n) \sim \) output tags, \( h = (h_1, \ldots, h_n) \sim \) encoder states, \( Z \) is a normalizer, such that the probabilities sum up to 1

\[ \psi \] A linear projection of \( h_i \),
the same as in standard labeling. No softmax here.

\[ \phi \] A table of transitions scores
between the tags \( \approx \) grammar of the tags.

\( P(y|h) \) is a probability distribution over the space of all possible tag sequences, not single tags.
CRF as a Trellis

Words $\rightarrow$

Label scores $\leftarrow$

$\psi_{0,0}$  $\psi_{1,0}$  $\psi_{2,0}$  $\psi_{3,0}$  $\psi_{4,0}$  $\psi_{5,0}$  $\psi_{6,0}$

$\psi_{0,1}$  $\psi_{1,1}$  $\psi_{2,1}$  $\psi_{3,1}$  $\psi_{4,1}$  $\psi_{5,1}$  $\psi_{6,1}$

$\psi_{0,2}$  $\psi_{1,2}$  $\psi_{2,2}$  $\psi_{3,2}$  $\psi_{4,2}$  $\psi_{5,2}$  $\psi_{6,2}$

$\psi_{0,3}$  $\psi_{1,3}$  $\psi_{2,3}$  $\psi_{3,3}$  $\psi_{4,3}$  $\psi_{5,3}$  $\psi_{6,3}$

$\psi_{0,4}$  $\psi_{1,4}$  $\psi_{2,4}$  $\psi_{3,4}$  $\psi_{4,4}$  $\psi_{5,4}$  $\psi_{6,4}$

$\psi_{0,5}$  $\psi_{1,5}$  $\psi_{2,5}$  $\psi_{3,5}$  $\psi_{4,5}$  $\psi_{5,5}$  $\psi_{6,5}$

Best labeling = finding best path in the graph
Making CRF tractable

😊 Good news: Everything is differentiable
😊 Bad news: There are exponentially many possible tag sequence \( y \).

1. Inference

Easy: we are only interested in the maximum \( \Rightarrow \) throw away \( Z \), throw away exponentiating, find maximum path in a trellis

2. Training

Tricky, we need the normalizer:

\[
Z = \sum_y \exp \left( \sum_{i=1}^{n} \psi(y_i|h_i) + \sum_{i=1}^{n-1} \phi(y_i, y_{i+1}) \right)
\]

Simple algebraic tricks allow dynamic programming algorithm.
**CRF Inference: Pseudocode**

\(\psi\) is a 2D array with labels scores, length \(\times\) labels of length \(T\) with \(n_{\text{labels}}\) labels

\(\phi\) label transition scores, shape: \(n_{\text{labels}} \times n_{\text{labels}}\)

---

```python
# 1. Search for the max-scoring path
scores = psi[0]
prev_pointers = []

for t in range(T):
    prev = []; new_scores = []
    for i in range(n_labels):
        cost_to_i = scores + phi[:, i] + psi[t, i]
        prev.append(cost_to_i.argmax())
        new_scores.append(cost_to_i.max())
    prev_pointers.append(prev)
    scores = new_scores
```

---

```python
# 2. Reverse-decode the path
# indices
best_path = [scores.argmax()]
for prev in reversed(prev_pointers):
    best_path.append(prev[best_path[-1]])
return reversed(best_path)
```
Factor out the last step to get a recurrent equation:

\[
\alpha_t(k) = \psi_k + \log \sum_i \exp(\alpha_{t-1}(j) + \phi_{i,k})
\]

\[
\alpha_0(k) = \psi_0(k)
\]

\[
Z = \log \sum_k \exp \alpha_T(k)
\]

There is an efficient implementation of log-sum-exp.

```python
alphas = psi[0]
for t in range(1, T):
    new_alphas = []
    for i in range(n_labels):
        new_alpha.append(psi[t, i] +
                         logsumexp(alpha[j] + phi[j, i]
                                    for j in range(n_labels)))
    alphas = new_alphas
return logsumexp(alphas)
```
Implementation in PyTorch

Package pytorch-crf

```bash
pip install pytorch-crf
```

Initialize the model:

```python
import torch
from torchcrf import CRF
num_tags = 5  # number of tags is 5
model = CRF(num_tags)
```

The module expects the unnormalized tag scores as the input.
Answer Span Selection
**Task:** Find an answer for a question given question in a coherent text.

http://demo.allennlp.org/machine-comprehension
Standard Dataset: SQuAD

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under *gravity*. The main forms of precipitation include drizzle, rain, sleet, snow, *graupel* and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals *within a cloud*. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
*gravity*

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
*graupel*

Where do water droplets collide with ice crystals to form precipitation?
*within a cloud*

- best articles from Wikipedia, of reasonable size (23k paragraphs, 500 articles)
- crowd-sourced more than 100k question-answer pairs
- complex quality testing (which got estimate of single human doing the task)

https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/

The same thing with BERT

Just throw everything into BERT: both the text and the question.

[CLS] ...question ... [SEP] ...the text with the answer ... [SEP]

Assign start the start and end labels.
Output Layer

1. Start-token probabilities: project each state to scalar $\rightarrow$ apply softmax over the context
2. End-token the same
3. At the end select the most probable span

A difference from standard labeling: scores get normalized:

- Standard: per label
- Here: over the entire text
### SQuAD Leaderboard

<table>
<thead>
<tr>
<th>method</th>
<th>Exact Match</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human performance</td>
<td>82.304</td>
<td>91.221</td>
</tr>
<tr>
<td>BiDAF trained from scratch</td>
<td>73.744</td>
<td>81.525</td>
</tr>
<tr>
<td>BERT</td>
<td>87.433</td>
<td>93.160</td>
</tr>
<tr>
<td>FPNet (Best in leaderboard; Feb 2021)</td>
<td>90.871</td>
<td>93.183</td>
</tr>
</tbody>
</table>

⚠️ Any time an NLP model is **better than humans**, something is **wrong**. Probably overfitting to specifics of the dataset.
Today’s Learning Outcomes

After this lecture you should be able to ...

1. Tell how various NLP tasks can be **formulated** as a **sequence labeling** problem
2. Reason about **problems with benchmarking** in NLP (using Visual Question Answering and Answer Span Selection as an Example)
3. Describe how **Named Entity Recognition** and **Answer Span Selection** solved using pre-trained language representations
Summary

1. **Named Entity Recognition**: a labeling problem with more clever training objective
2. **Answer Span Selection**: showcasing the strength of Transformers, in the end labeling problem too

http://ufal.cz/courses/npfl1124