Deep Learning Applications in Natural Language Processing

Jindřich Libovický

April 19, 2023
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, Answer Span Selection and Visual Question Answering can be solved using pre-trained language representations
Outline

Named Entity Recognition

Answer Span Selection

Joint Modeling of Language and Vision
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.

## Entity Types

Different entity recognizers use different sets. Example:

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

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:

There are over 1000 compositions by Johan Sebastian Bach.

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
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.
psi is a 2D array with labels scores, length × labels of length T with n_labels labels

phi label transition scores, shape: n_labels × n_labels

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

for t 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

# 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)
CRF Training: Compute the normalizer

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

g 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 modul 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/

Challenges of the task

Two inputs: The question and the text containing the answer.

1. Before Transformers
   A super complicated architecture to reasonably combine both inputs.
   A chicken-egg problem: what to process first?

2. With Transformers and BERT
   Self-attention compares everything with everything.
   Gets reduced to labeling problem.
1. Get text and question representation from using your favourite architecture.
2. Compute a similarity between all pairs of words in the text and in the question.
3. Collect all informations we have for each token.
4. Classify where the span is.

BiDAF: Image

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.
1. Start-token probabilities: project each state to scalar → 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.
Joint Modeling of Language and Vision
Deep learning is a dominant methodology in computer vision too

- Language representations = vectors
- Image representations = vectors

The vision & language research tries to align the representations

**Tasks:** Image retrieval by caption, image captioning, visual question answering, visual navigation
2D Convolution over an Image

Basic method in deep learning for computer vision.

RGB image $9 \times 9 \times 3$

convolutional map $4 \times 4 \times 6$

stride 2
kernel size 3
filter size 6
• Architecture: convolutions, max-pooling, residual connections, batch normalization, 50–150 layers
• Most frequent pre-training: ImageNet — millions of images, 1k labels
Object Detection: Faster R-CNN

- Representation from pre-trained network for image classification
- Label areas with possible objects (proposals)
- Filter using a classifier

VilBERT: Joint Language-and-Vision Pre-training

Trained on image-caption pairs.

- Predict what was masked on the input: the same as BERT
- Mask image object – predict what was missing ⇒ needs to look it up in the text
- Words in the caption can be aligned with the image objects
- Matching words and objects – another training objective

GQA
1. What is the woman to the right of the boat holding? umbrella
2. Are there men to the left of the person that is holding the umbrella? no
3. What color is the umbrella the woman is holding? purple

GQA
1. Is that a giraffe or an elephant? giraffe
2. Who is feeding the giraffe behind the man? lady
3. Is there any fence near the animal behind the man? yes
4. On which side of the image is the man? right
5. Is the giraffe is behind the man? yes

It’s a classification problem

Use the ViLBERT representation and train a classifier predicting a single word.

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, Answer Span Selection and Visual Question Answering can be 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

3. **Vision and language**: The same deep learning methods can be used to align vision and language representations