Deep Learning Applications in Natural Language Processing

Jindřich Libovický

May 6, 2021
Named Entity Recognition

Answer Span Selection

Unsupervised Dictionary Induction

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:

- **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**: 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:

There are over 1000 compositions by Johan Sebastian Bach.

Special B and I tags for each of the entity types.
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.
• 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
  \[\Rightarrow \text{Conditional Random Fields (CRF)}\] 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 ≈ grammar of the tags.

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

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

Algebraic tricks: start decomposing by the first tags, factor it out, simplify, etc. \( \Rightarrow \) leads to dynamic programing algorithm.
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

- 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.

Attention Flow

\[ S_{ij} = w^T [h_i, c_j, h_i \odot c_j] \]

Captures affinity / similarity between pairs of question and context words.
Context-to-query Attention

\[ u_1 \; u_2 \; u_3 \; u_4 \; u_5 \]

query

\[ h_1 \; h_2 \; h_3 \; h_4 \; h_5 \; h_6 \; h_7 \; h_8 \; h_9 \; h_{10} \; h_{11} \; h_{12} \]

class \( H \)

\[ \tilde{u}_1 \; \tilde{u}_2 \; \tilde{u}_3 \; \tilde{u}_4 \; \tilde{u}_5 \; \tilde{u}_6 \; \tilde{u}_7 \; \tilde{u}_8 \; \tilde{u}_9 \; \tilde{u}_{10} \; \tilde{u}_{11} \; \tilde{u}_{12} \]

\[ \times \]

softmax

\[ \times \]

weighted sum
Query-to-Context Attention

\[ u_1 \quad u_2 \quad u_3 \quad u_4 \quad u_5 \]

context \( H \)

\[ h_1 \quad h_2 \quad h_3 \quad h_4 \quad h_5 \quad h_6 \quad h_7 \quad h_8 \quad h_9 \quad h_{10} \quad h_{11} \quad h_{12} \]

weighted sum

maximum

Named Entity Recognition  Answer Span Selection  Unsupervised Dictionary Induction  Joint Modeling of Language and Vision
Modeling Layer

- concatenate: LSTM outputs for each context word, context-to-query-vectors
- copy query-to-context vector to each of them
- apply one non-linear layer and bidirectional LSTM
Output Layer

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**
Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California.

As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.
There are 13 natural reserves in Warsaw—among others, Bielany Forest, Kabaty Woods, Czerniaków Lake. About 15 kilometres (9 miles) from Warsaw, the Vistula river's environment changes strikingly and features a perfectly preserved ecosystem, with a habitat of animals that includes the otter, beaver and hundreds of bird species. There are also several lakes in Warsaw—mainly the oxbow lakes, like Czerniaków Lake, the lakes in the Łazienki or Wilanów Parks, Kamionek Lake. There are lot of small lakes in the parks, but only a few are permanent—the majority are emptied before winter to clean them of plants and sediments.
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]

BERT Encoder

Assign start the start and end labels the same as before.
<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.
Unsupervised Dictionary Induction
**Task:** Get a translation dictionary between two languages using monolingual data only.

- makes NLP accessible for low-resourced languages
- basic for unsupervised machine translation
- hot research topic (at least 10 research papers on this topic this year)

How it is done

1. Train word embeddings on large monolingual corpora.
2. Find a mapping between the two languages.

So far looks simple...
Dictionary and Common Projection

$X$, $Z$ embedding matrices for 2 languages.
Dictionary matrix $D_{ij} = 1$ if $X_i$ is translation of $Z_j$.

Supervised projection between embeddings
Given existing dictionary $D$ (small seed dictionary):

$$\arg\max_{W_Z, W_X} \sum_i \sum_j D_{ij} \cdot \text{similarity} (X_i; W_X, Z_j; W_Z) \left( X_i W_X (Z_j; W_Z)^T \right)$$

...but we need to find all $D$, $W_X$, and $W_Z$. 
Question: How would you interpret this matrix?
It is a table of similarities between pairs of words.
If the Vocabularies were Isometric...

- $M_X = XX^T$ and $M_Z = ZZ^T$ would only have permuted rows and columns
- if we sorted values in each row of $M_X$ and $M_Z$, corresponding words would have the same vectors

Let’s assume, it is true (at least approximately)

\[
D_{i,:} \leftarrow 1 \begin{bmatrix} \argmin_j (M_X)_{i,:} (M_Z)_{j,:}^T \end{bmatrix}
\]

Assign nearest neighbor from the other language.

......in practice tragically bad but at least good initialization.
Self-Learning

Iterate until convergence:

1. Optimize $W_Z$ and $W_X$, w.r.t to current dictionary

$$\text{argmax}_{W_Z, W_X} \sum_i \sum_j D_{ij} \cdot (X_i W_X (Z_j W_Z)^T)$$

2. Update dictionary matrix $D$

$$D_{ij} = \begin{cases} 1, & \text{if } i \text{ is nearest neighbor of } j \text{ or vice versa} \\ 0, & \text{otherwise} \end{cases}$$
## Accuracy on Large Dictionary

<table>
<thead>
<tr>
<th>Supervision</th>
<th>Method</th>
<th>EN-IT</th>
<th>EN-DE</th>
<th>EN-FI</th>
<th>EN-ES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mikolov et al. (2013)</td>
<td>34.93†</td>
<td>35.00†</td>
<td>25.91†</td>
<td>27.73†</td>
</tr>
<tr>
<td>5k dict.</td>
<td>Faruqui and Dyer (2014)</td>
<td>38.40*</td>
<td>37.13*</td>
<td>27.60*</td>
<td>26.80*</td>
</tr>
<tr>
<td></td>
<td>Shigeto et al. (2015)</td>
<td>41.53†</td>
<td>43.07†</td>
<td>31.04†</td>
<td>33.73†</td>
</tr>
<tr>
<td></td>
<td>Dinu et al. (2015)</td>
<td>37.7</td>
<td>38.93*</td>
<td>29.14*</td>
<td>30.40*</td>
</tr>
<tr>
<td></td>
<td>Lazaridou et al. (2015)</td>
<td>40.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Xing et al. (2015)</td>
<td>36.87†</td>
<td>41.27†</td>
<td>28.23†</td>
<td>31.20†</td>
</tr>
<tr>
<td></td>
<td>Zhang et al. (2016)</td>
<td>36.73†</td>
<td>40.80†</td>
<td>28.16†</td>
<td>31.07†</td>
</tr>
<tr>
<td></td>
<td>Artetxe et al. (2016)</td>
<td>39.27</td>
<td>41.87*</td>
<td>30.62*</td>
<td>31.40*</td>
</tr>
<tr>
<td></td>
<td>Artetxe et al. (2017)</td>
<td>39.67</td>
<td>40.87</td>
<td>28.72</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Smith et al. (2017)</td>
<td>43.1</td>
<td>43.33†</td>
<td>29.42†</td>
<td>35.13†</td>
</tr>
<tr>
<td></td>
<td>Artetxe et al. (2018a)</td>
<td>45.27</td>
<td>44.13</td>
<td></td>
<td>32.94</td>
</tr>
<tr>
<td>25 dict.</td>
<td>Artetxe et al. (2017)</td>
<td>37.27</td>
<td>39.60</td>
<td>28.16</td>
<td>-</td>
</tr>
<tr>
<td>Init. heurist.</td>
<td>Smith et al. (2017), cognates</td>
<td>39.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Artetxe et al. (2017), num.</td>
<td>39.40</td>
<td>40.27</td>
<td>26.47</td>
<td>-</td>
</tr>
<tr>
<td>None</td>
<td>Zhang et al. (2017a), $\lambda = 1$</td>
<td>0.00*</td>
<td>0.00*</td>
<td>0.00*</td>
<td>0.00*</td>
</tr>
<tr>
<td></td>
<td>Zhang et al. (2017a), $\lambda = 10$</td>
<td>0.00*</td>
<td>0.00*</td>
<td>0.01*</td>
<td>0.01*</td>
</tr>
<tr>
<td></td>
<td>Conneau et al. (2018), code†</td>
<td>45.15*</td>
<td>46.83*</td>
<td>0.38*</td>
<td>35.38*</td>
</tr>
<tr>
<td></td>
<td>Conneau et al. (2018), paper†</td>
<td>45.1</td>
<td>0.01*</td>
<td>0.01*</td>
<td>35.44*</td>
</tr>
<tr>
<td></td>
<td>Proposed method</td>
<td><strong>48.13</strong></td>
<td><strong>48.19</strong></td>
<td>32.63</td>
<td><strong>37.33</strong></td>
</tr>
</tbody>
</table>
Try it yourself!

- Pre-train monolingual word embeddings using FastText / Word2Vec
- Install VecMap
  
  https://github.com/artetxem/vecmap

  python3 map_embeddings.py --unsupervised SRC.EMB TRG.EMB SRC_MAPPED.EMB TRG_MAPPED.EMB
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
Convolutinal Network for Image Classification

- **Architecture**: convolutions, max-pooling, residual connections, batch normalization, 50–150 layers
- **Most frequent pre-training**: ImageNet — millions of images, 1k labels
• Representation from pre-trained network for image classification
• Label areas with possible objects (proposals)
• Filter using a classifier

Object Detection: Faster R-CNN

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

It’s a classification problem

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. **Dictionary induction**: create a dictionary from embeddings only

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

http://ufal.cz/courses/npfl124