Benjamin Roth

Deep learning for natural language processing

Workshop @ The Digital Product School / UnternehmerTUM
5.3.2018
From unstructured to structured data

• About me
  • Research focus: Information extraction (IE) from text
  • Currently **Vertretung** for Hinrich Schütze in Munich
• Most information about real world is unstructured.
  • “At the age of 19, Martin Luther entered the University of Erfurt.”
  • “On 2 July 1505 he was returning to Erfurt after visiting his parents in Mansfeld.”

⇒ Did Martin Luther live in Erfurt?

• Turning unstructured data into structured form:
  Automated knowledge base population (KBP)

⇒ lived_in(M_Luther, Erfurt) 0.8942
Why more structured data?

- Algorithms need structured data with specific interpretation
  - Databases, triple stores, ...
  - Hadoop, Spark, ...
- Computer science ↔ other disciplines:
  - Computational social science: Detecting real world conflict and political events [O’Connor, 2013]
  - Bio-informatics: Extracting genome and protein interactions from research publications [Segura-Bedmar et al., 2013]
  - Market research: Extracting typical use-cases of food and products [Wiegand et al., 2012].
  - ...
Deep Learning for NLP
What is machine learning?

**Supervised** learning:

- ``Given X predict Y’’
- ``What is input, what is output?‘’
- Most common setting in machine learning
- Needs training data with known output
- **Example:** Is an email (``input‘‘) spam or not (``output‘‘)?

```
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

SPAM

SPAM

NO_SPAM
What is machine learning?

Unsupervised learning:

• **Find structure in data** (e.g. groups of similar items)
• Only ``input`` needed, no ``output``
• Useful for helping supervised tasks, or for human data exploration
• **Example**: Find groups of similar emails

Example diagrams:

- EMAILS_ABOUT_ONLINE_PHARMACIES
- EMAILS_ABOUT_DEADLINES
- EMAILS_ABOUT_SPORT_EVENTS
**Feature-based learning**

- Input representation: explicit set of features (e.g. set words in an email)
- Learn a **prediction rule that operates directly on features**
- Features themselves are not learned
- Prediction rule often **linear**

**Linear:**

- “More of this feature \(\rightarrow\) More of that output”
- Cannot model interactions between features
- Cannot model saturation of features
- Perceptron, linear SVM, logistic regression, classical CRF,...
Representation learning = Deep Learning = Neural Networks

- **Raw input** instead of defined feature representation:
  - Text: Sequence of words or characters
  - Images: Pixels
- **Learn higher-level abstractions**
Representation learning = Deep Learning = Neural Networks

• **Raw input** instead of defined feature representation:
  • Text: Sequence of words or characters
  • Images: Pixels

• **Learn higher-level abstractions**
  • **Non-linear functions** can model interactions of lower-level representations
  • E.g.:
    ``
    ``The plot was **not** particularly **original.**’’  → **negative** movie review
  • Typical setup for natural language processing (NLP)
    • Model starts with learned representations for words
      → **word vectors**
    • Word vectors are combined to represent larger units (sentences, documents)
Word Vectors = Embeddings

Definition

The embedding of a word $w$ is a dense vector $\vec{v}(w) \in \mathbb{R}^k$ that represents semantic and other properties of $w$. Typical values are $50 \leq k \leq 1000$.

<table>
<thead>
<tr>
<th>$U$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>-0.44</td>
<td>-0.30</td>
<td>0.57</td>
<td>0.58</td>
<td>0.25</td>
</tr>
<tr>
<td>boat</td>
<td>-0.13</td>
<td>-0.33</td>
<td>-0.59</td>
<td>0.00</td>
<td>0.73</td>
</tr>
<tr>
<td>ocean</td>
<td>-0.48</td>
<td>-0.51</td>
<td>-0.37</td>
<td>0.00</td>
<td>-0.61</td>
</tr>
<tr>
<td>wood</td>
<td>-0.70</td>
<td>0.35</td>
<td>0.15</td>
<td>-0.58</td>
<td>0.16</td>
</tr>
<tr>
<td>tree</td>
<td>-0.26</td>
<td>0.65</td>
<td>-0.41</td>
<td>0.58</td>
<td>-0.09</td>
</tr>
</tbody>
</table>
Word vectors – regularities in vector space (2D projection)
Word vectors – regularities in vector space (2D projection)
Word vectors – regularities in vector space (2D projection)
Machine Learning as a black box

- ``Given X predict Y``
- ``What is input, what is output?``
- What is input?
  - Text: a sequence of tokens (sentence or document)
- What is output?
Machine Learning as a black box

- What is input?
  - Text: a sequence of tokens (sentence or document)

- What is output?
  - One of several categories (Classification)
    → Spam / no spam
  - A numerical value (Regression)
    → Number of stars given a review
  - A prediction for each token (Tagging)
    → Mark each word that is a person, location or organization
  - Another text (Sequence-to-sequence)
    → Translation of sentence into a different language
Deep learning and modularization

- Deep learning models for NLP
  - Modul 1: **Encode sentence**
    - Result: Vector representation with learned features
  - Modul 2: **Make prediction**
    - Input: Learned Features
- **Deep learning provides an API for machine learning**
  - A main advantage, even if sometimes traditional models perform equally well
- **Interfaces are learned vector representation**
  - input $\rightarrow$ vector(s)
  - vector(s) $\rightarrow$ vector(s)
  - vector(s) $\rightarrow$ output
Layers

- A neural network consists of different layers: Mappings from vectors to vectors
- The output of one layer is the input to the next layer
- Input and output dimensions do not need to match
- First layer is word vector lookup
- Last layer is prediction

Prediction / Output

Layer 3

Layer 2

Layer 1

Instance / input representation
Layers in a neural network for sentiment prediction

the sopranos was the best show
Recurrent Neural Networks (RNNs)

- A sentence is recursively summarized by a non-linear function that combines current word vector with the summary at the previous position.

The most popular variant of RNNs is the LSTM (Long Short-Term Memory network).
It has advantages when training with long sequences.
Common non-linearities

- tanh

- ReLu

- logistic sigmoid

- softmax

\[ y_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \]
Common non-linearities

- tanh
- ReLu
- logistic sigmoid
- softmax

used for last layer / prediction

\[
y_i = \frac{e^{x_i}}{\sum_j e^{x_j}}
\]
The state of the art for many applications: Bidirectional LSTMs

the sopranos was the best show
Convolutional Neural Networks (CNNs)

The Sopranos was the best show.
Deep learning and modularization

- General purpose mechanisms ... ... independent of specific problem
- Optimal parameters learned from a task-specific training corpus
- For example: **Mechanisms to encode a sequence**
  - Recurrent Neural Networks (RNN, LSTM/GRU, QLSTM/QGRU)
  - Convolutional Neural Networks (CNNs)
  - Self-Attention, ...
- **Mechanism to produce an output** depend on the task
  - E.g. multiclass prediction: Softmax
  - E.g. tagging: Neural Conditional Random Fields
  - ...
Deep learning and modularization

• Interfaces are learned vector representation
  • input $\rightarrow$ vector
  • vector(s) $\rightarrow$ vector
  • sector $\rightarrow$ output
• Learned vector representations as the universal "language" of neural networks
• Makes it easy to
  • Combine different input modalities (e.g. audio+video+subtitles)
  • Pre-train parts of the architecture (e.g. word vectors, sentence encoder)
  • Predict different outputs from the same representation
Deep learning and modularization

Deep learning provides a modular way of structuring your problem.
Example Problem: Question Answering

Text: “Yesterday, Emanuel Macron and his wife Brigitte Trogneux visited the Louvre Abu Dhabi Museum”

Question: “Who is Emanuel Macron married to?”

Answer: “Brigitte Trogneux”

• **What is input?** How to encode input?
• **What is output?** Task type?
  • Classification?
  • Regression?
  • Tagging?
  • Sequence-to-sequence?
Example Problem: Question Answering

Text: “Yesterday, Emanuel Macron and his wife Brigitte Trogneux visited the Louvre Abu Dhabi Museum”

Question: “Who is Emanuel Macron married to?”

Answer: “Brigitte Trogneux”

• What is input?
  • Question + sentence

• How to encode input?
  • Concatenate question and sentence
    Who is Emanuel Macron married to? # Yesterday, Emanuel Macron and his wife Brigitte Trogneux visited the Louvre Abu Dhabi Museum
  • Encode question, encode sentence, and concatenate vectors
  • Attention mechanisms (BiDAF), …
Example Problem: Question Answering

Question: “Who is Emanuel Macron married to?”

- What is output?
  - Answer = substring of text
    *Yesterday, Emanuel Macron and his wife Brigitte Trogneux visited the Louvre Abu Dhabi Museum*

- Task type?
  - Predict start and end positions of answer.
    (Classification/Regression)
Example Problem: Question Answering

Question: ‘Who is Emanuel Macron married to?’

• What is output?
  • Answer = substring of text
    *Yesterday, Emanuel Macron and his wife Brigitte Trogneux visited the Louvre Abu Dhabi Museum*

• Task type?
  • For combinations of start and end positions, predict whether subspan is answer. (Classification)
Example Problem: Question Answering

Question: “Who is Emanuel Macron married to?”

• What is output?
  • Answer = substring of text
    *Yesterday, Emanuel Macron and his wife Brigitte Trogneux visited the Louvre Abu Dhabi Museum*

• Task type?
  • For each word, mark whether it belongs to the answer (tagging)
Example Problem: Question Answering

Question: "Who is Emanuel Macron married to?"

- What is output?
  - Answer = substring of text
    "Yesterday, Emanuel Macron and his wife Brigitte Trogneux visited the Louvre Abu Dhabi Museum"

- Task type?
  - "Translate" question+text into answer (sequence-to-sequence)

  "Who is Emanuel Macron married to? # Yesterday, Emanuel Macron and his wife Brigitte Trogneux visited the Louvre Abu Dhabi Museum"
  → Brigitte
  → Trogneux
  → <END>
Deep Learning Frameworks

- Specify the model
- Optimize parameters (training)
- Make predictions
- Deploy training and prediction
3 Deep Learning Frameworks (Python)

- TensorFlow (2015-)
  - Developed by Google
  - *Static computation graph*: model specification $\rightarrow$ compilation $\rightarrow$ training/running/debugging
  - Strengths:
    - Industrial strength deployment options
    - Large community / strong backing
- Keras (2015-)
  - High-level deep learning abstractions
  - Takes away 95% of programming overhead (and some flexibility)
  - Great way to start for standard problems (classification, tagging,...)
  - Since 2017 integrated into TensorFlow core
3 Deep Learning Frameworks (Python)

- Pytorch (2016-)
  - Developed by Facebook AI
  - Dynamic computation graph:
    model specification = model → training/running/debugging
  - Great for prototyping of novel model types
    - Easy to integrate control flow logic (hierarchical models, reinforcement learning,...)
    - Meaningful debugging output
- There are many more: Theano, CNTK, MXNet, Caffe, ...
Building neural networks with Keras

```python
from keras.models import Sequential
from keras.layers import *
model = Sequential()
model.add(Embedding(vocabulary_size, 100))
model.add(Bidirectional(LSTM(100)))
model.add(Dense(1))
model.compile(optimizer='adam',
              loss='binary_crossentropy')
model.fit(x_train, y_train)
```
Deep Learning

What society thinks I do
What my friends think I do
What other computer scientists think I do

What mathematicians think I do
What I think I do
What I actually do

In [1]:
import keras
Using TensorFlow backend.
**Q:** I want to use deep learning for NLP. Where do I start?

1. **Make sure your problem fits into the scheme**
   `Given X predict Y` (What is input? What is output?)

2. **Get training data**, i.e. input-output pairs
   - Input alone is not sufficient!
   - Collect data (e.g. from observed user behavior)
   - Annotation, crowd-sourcing (Amazon Mechanical Turk) necessary?
   - Automatic labelling possible? (by combining data sources)
   - Rule of thumb: for NLP **at least 10000 training instances**
     (better: several millions)

3. **Split training data** into three parts
   - Training (80%): used by model training to optimize parameters
   - Development (10%): for monitoring effect of changes to architecture
   - Test (10%): used to detect overfitting on development data
**Q: I want to use deep learning for NLP. Where do I start?**

4. **Determine task type**
   - Classification?
   - Regression?
   - Tagging?
   - Sequence-to-sequence?

5. **Choose deep learning framework**
   - My recommendation: start with Keras

6. **Build architecture**
   - Encode text with *bidirectional LSTMs*
   - Encode images with pre-trained architecture (e.g. VGG+Imagenet)
   - Encode simple additional input with embeddings

7. **Iterate and improve architecture so that performance on development data increases**
Q: I want to use deep learning for NLP. Where do I start?

4. Determine **task type**
   - Classification?
   - Regression?
   - Tagging?
   - Sequence-to-sequence?

5. Choose **deep learning framework**
   - My recommendation: start with Keras

6. Build architecture
   - Encode text with **bidirectional LSTMs**
   - Encode images with pre-trained architecture (e.g. VGG+Imagenet)
   - Encode simple additional input with embeddings

7. Iterate and improve architecture so that performance on development data increases

Thank You! Any Questions?