Deep learning landscape

Computer Vision
Example Project

Natural Language Processing (NLP)
Example Project
Sample GNN use case
## Sample GNN use case

![Image of a receipt from Dewey Hobbs GmbH]

### Rechnung

- **Rechnungs-Nr.**: 6000245-1001
- **Bestellung-Nr.**: KA12345
- **Kunden-Nr.**: 1001

**Datum**: 07/01/2020

<table>
<thead>
<tr>
<th>Pos</th>
<th>Leistung</th>
<th>MwSt.</th>
<th>Einzelpreis</th>
<th>Anzahl</th>
<th>Gesamtpreis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DGN-Bohrmittel</td>
<td>19 %</td>
<td>19,91 EUR</td>
<td>5</td>
<td>99,55 EUR</td>
</tr>
<tr>
<td>2</td>
<td>DGN-Bohrmittel</td>
<td>19 %</td>
<td>22,00 EUR</td>
<td>1</td>
<td>22,00 EUR</td>
</tr>
<tr>
<td>3</td>
<td>DGN-Bohrmittel</td>
<td>19 %</td>
<td>66,00 EUR</td>
<td>1</td>
<td>66,00 EUR</td>
</tr>
</tbody>
</table>

**abc:**

- **ABC Chemicals AG**
- **Kochstraße 2**
- **5678 Salzstadt**

**Dewey Hobbs GmbH**

- **Deweystraße 21**
- **12345 Hobbshausen**

- **Tel.: 0211 12345 67**
- **E-Mail: accounting@hobbs.de**
- **Internet: www.deweyhobbs.de**
A typical information extraction pipeline

- Encode information in a graph
- Use graph algorithms to process the information

Where do graphs enter in this picture?
A typical information extraction pipeline

Where do graphs enter in this picture?
- Encode information in a graph
- Use graph algorithms to process the information

Agenda:
- Graphs and GNNs
- Comparison with convolutional networks
- Use case
What is a graph?
What is a graph?

Variations: simple graph, multigraph, pseudograph; directed or not
What is a graph?

Definition: A directed pseudograph, or simply graph, is
- a set $V$ of vertices
- a set $E$ of edges
- functions source, target: $E \rightarrow V$
**What is a graph?**

**Definition:** A *directed pseudograph*, or simply *graph*, is:
- a set $\mathcal{V}$ of vertices
- a set $\mathcal{E}$ of edges
- functions $\text{source}, \text{target} : \mathcal{E} \rightarrow \mathcal{V}$

**Example:**
- $\mathcal{V} = \{A, B, C, ..., E\}$
- $\mathcal{E} = \{p, q, r, ..., v\}$
- $\text{source}(p) = A$
- $\text{target}(p) = B$
- $\text{source}(v) = E$
- $\text{target}(v) = E$
What is a graph?

**Definition:** A *directed pseudograph*, or simply *graph*, is
- a set $\mathcal{V}$ of vertices
- a set $\mathcal{E}$ of edges
- functions source, target: $\mathcal{E} \rightarrow \mathcal{V}$

**Example:**
- $\mathcal{V} = \{A, B, C, ..., E\}$
- $\mathcal{E} = \{p, q, r, ..., v\}$
- $\text{source}(p) = A$
- $\text{target}(p) = B$
- $\text{source}(v) = E$
- $\text{target}(v) = E$

**Note:** Often there are labels associated to vertices and edges.
- $L_0: \mathcal{V} \rightarrow \{\text{vertex labels}\}$
- $L_1: \mathcal{E} \rightarrow \{\text{edge labels}\}$
Representing graphs in Python
Representing graphs in Python

adjacencies = [
    (0, 1),  # e_0
    (0, 1),  # e_1
    (1, 2),  # e_2
    ...,  # e_2
    (4, 4),  # e_6
]

Representing graphs in Python

A graph with nodes labeled by NodeT and edges labeled by EdgeT can be modeled as follows:

```python
class Graph(Generic[NodeT, EdgeT]):
    nodes: List[NodeT]
    edges: List[EdgeT]
    adjacencies: List[Tuple[int, int]]

    def source(self, k: int) -> int:
        """Source of the kth edge""
        return self.adjacencies[k][0]

    def target(self, k: int) -> int:
        """Target of the kth edge""
        return self.adjacencies[k][1]
```

```
adjacencies = [
    (0, 1),  # e_0
    (0, 1),  # e_1
    (1, 2),  # e_2
    ...,
    (4, 4),  # e_6
]
```
Graph neural networks

Let’s consider the node classification problem:

- Input: a graph $G$ of type $\text{Graph}[\text{Tensor}, \text{EnumT}]$
- Output: a labeling of its vertices, $L: \mathcal{V} \rightarrow \{0, 1\}$

labels = [0, 1, 0, 0, 1, 0, 0, 1]
For instance:

\[ y_v = \sigma \left( \sum_{e: v \rightarrow w} \frac{1}{\text{deg}_t^+(v)} W_{t(e)} x_w \right) \]

where

- \( y_v \) is the output feature vector of node \( v \)
- \( x_w \) is the input feature vector of node \( w \)
- \( t(e) \) is the type of the edge \( e: v \rightarrow w \)
- \( W_t \) is a learned weight matrix corresponding to edge type \( t \)
- \( \text{deg}_{t}^+(v) \) is the number of incoming edges of type \( t \) for node \( v \)
- \( \sigma \) is an activation function

Schlichtkrull et al, “Modeling Relational Data with Graph Convolutional Networks”, arxiv:1703.06103
Comparison with convolutional networks

Graph (convolutional) networks generalize CNNs from computer vision

\[ y_{i,j} = \sigma \left( \sum_{k=-1,0,1}^{k=-1,0,1} \sum_{l=-1,0,1}^{l=-1,0,1} W_{k,l} x_{i+k,j+l} \right) \]

where

- \( y_{i,j} \) is the output feature vector of pixel \((i, j)\)
- \( x_{i,j} \) is the input feature vector of pixel \((i, j)\)
- \( W_{k,l} \) are learned weight matrices
- \( \sigma \) is an activation function
Comparison with convolutional networks

Graph (convolutional) networks generalize CNNs from computer vision

\[ y_{i,j} = \sigma \left( \sum_{k=-1,0,1}^{l=-1,0,1} W_{k,l} x_{i+k,j+l} \right) \]

where

- \( y_{i,j} \) is the output feature vector of pixel \((i, j)\)
- \( x_{i,j} \) is the input feature vector of pixel \((i, j)\)
- \( W_{k,l} \) are learned weight matrices
- \( \sigma \) is an activation function
Comparison with convolutional networks

Graph (convolutional) networks generalize CNNs from computer vision

\[ y_p = \sigma \left( W_O x_p + \sum_{q \rightarrow p} W_t x_q \right) \]

where

- \( x_p \) is the input feature vector of pixel \( p \)
- \( y_p \) is the output feature vector of pixel \( p \)
- \( t \in \{N, NW, W, \ldots\} \) is a cardinal direction
- \( W_t \) are learned weight matrices
- \( \sigma \) is an activation function
Comparison with convolutional networks

A graph representing the entire image
Use case: information extraction from tables

Idea: Model the document structure as a graph

- each word is a node in the graph
- neighbouring words are connected by edges
Use case: information extraction from tables

Nodes:
- Each word is a node in the graph
Use case: information extraction from tables

Nodes:
- Each word is a node in the graph

Edges:
- Neighboring words are connected
Use case: information extraction from tables

Nodes:
- Each word is a node in the graph

Edges:
- Neighboring words are connected

<table>
<thead>
<tr>
<th>Seller</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>We-Supply</td>
<td>Delivery Address</td>
</tr>
<tr>
<td>NC Suppliers Corporation</td>
<td>ACME US&amp;A 999 Supreme Industrial Road</td>
</tr>
<tr>
<td>P.O. Box 123456</td>
<td>Anderson</td>
</tr>
<tr>
<td>Atlanta</td>
<td>South Carolina</td>
</tr>
<tr>
<td>GA 12345-98765</td>
<td>12345</td>
</tr>
<tr>
<td>UNITED STATES</td>
<td>UNITED STATES</td>
</tr>
<tr>
<td>Phone: 111-222-3334</td>
<td>Fax</td>
</tr>
<tr>
<td>Ship Via</td>
<td>Payment Terms</td>
</tr>
<tr>
<td>Road</td>
<td>1 Day/Net</td>
</tr>
<tr>
<td>Terms Of Delivery</td>
<td>Delivery Date</td>
</tr>
<tr>
<td>Free on board</td>
<td>3/31/19</td>
</tr>
</tbody>
</table>


For orders placed by AMCE US&A USA Corporation, the AMCE US&A USA Corporation Standard Conditions of
Use case: a possible model architecture

Legend

W: Number of words
S: Number of static text features
C: Number of classes
⊕: Concatenate tensors
Use case: results

About 1000 training documents

Macro F1 score: 88%
Implementations, literature

- **PyTorch Geometric**
  - “GNN Cheatsheet” in the docs contains an interesting list of papers
- **Spektral** for TensorFlow
- **Deep Graph Library** (framework agnostic)
- **https://en.wikipedia.org/wiki/Graph_neural_network** exists since 5 July 2021