Apprendere dai Dati Sequenziali

Tecniche e Applicazioni

Flavio Giobergia
Politecnico di Torino

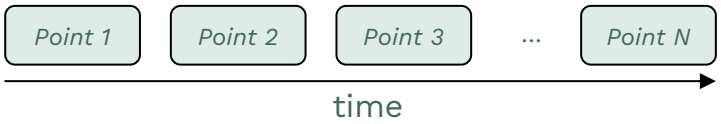
April 5, 2024
AFC Digital Hub

Table of Contents

Sequential data

Sequential data

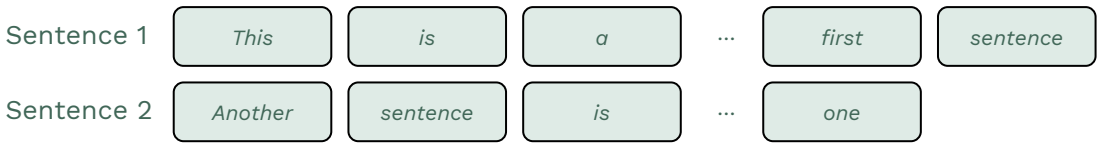
- Sequential data is data that is inherently ordered
  - Generally, by time
- The order of the data points is significant, and often carries information essential for *analysis* and *prediction*
- *Temporal dependency*
  - i^{th} point generally depends on 1^{st}, 2^{nd}, ..., $(i-1)^{th}$
  - What surrounds i^{th} point provides its context



- Variable length
  - Sequential data can vary in length from sequence to sequence
  - Unlike traditional datasets!

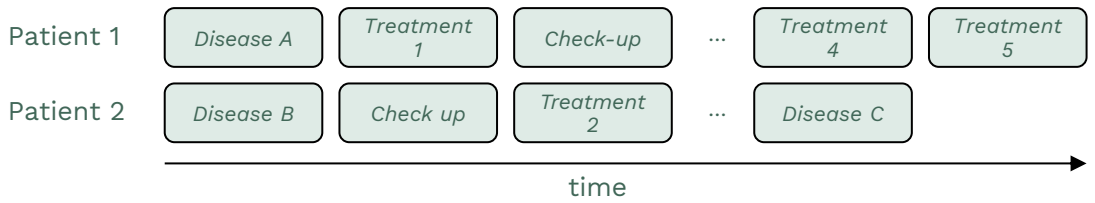
Sequences across Domains

- **Natural Language Processing (NLP)**
  - From text generation to sentiment analysis, understanding the sequence of words is crucial for context and meaning.



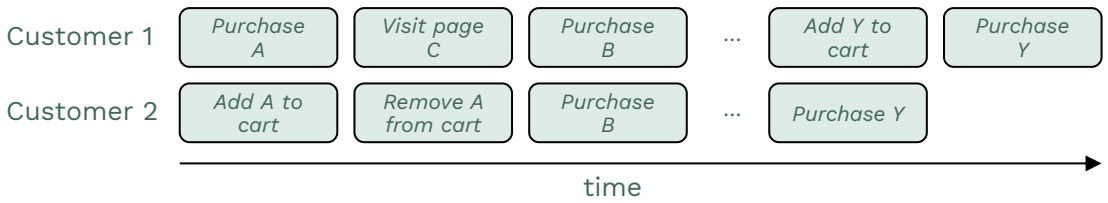
Sequences across Domains

- **Healthcare**
  - Patient records, treatment sequences, and disease progression studies utilize sequential data for better outcomes.



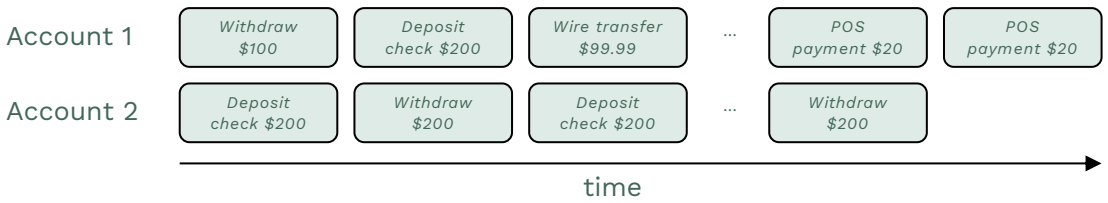
Sequences across Domains

- **E-commerce and Retail**
  - Analyzing customer transaction sequences to understand purchasing behavior and personalize recommendations.



Sequences across Domains

- **Finance & banking**
  - Transactions, stock prices, market trends, and economic indicators all rely on historical data sequences for forecasting.



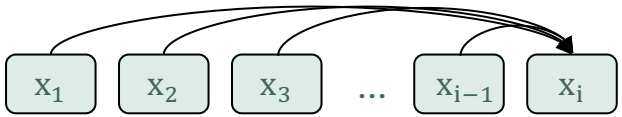
Sequential tasks

- **Prediction and Forecasting**
  - Enables the prediction of future events based on past sequences (what will the customer purchase next?)
- **Sequence Generation**
  - Producing new sequences that resemble the given data, such as generating synthetic text, or sequences of user behavior or financial transactions.
- **Classification**
  - Assigning each sequence or subsequences to a category or class, such as classifying the sentiment of a snippet of text.
- **Clustering**
  - Grouping sequences that are similar to each other but not pre-labeled. This can be used for customer segmentation based on spending patterns.
- **Anomaly Detection**
  - Finding patterns in the data that do not conform to expected behavior. This is often used for fraud detection or system health monitoring.

Sequential Data Challenges

- **Variable Length**
  - Conventional machine learning models require fixed-length inputs!
  - Specific techniques need to be used
  - or, information needs to be pooled (aggregated)

- **Temporal Dependencies**
  - Each point depends on previous ones!
    - $P(x_i) = P(x_i|x_{i-1}, x_{i-2}, \ldots, x_1)$
  - Capturing these (possibly long-term) dependencies is critical for many applications.



- **Need for Memory in Models**
  - Sequential models need to remember previous inputs, which can be challenging for certain algorithms.
  - Memory mechanisms (e.g. in RNNs, LSTMs) are used to selectively remember or forget information over time

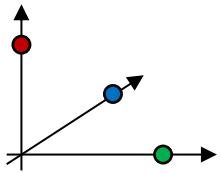
Word embeddings

Why word embeddings?

- Words are not numbers!
  - And ML algorithms generally work with numbers!
- Naïve approaches (e.g., 1-hot encoding) do not capture semantic similarity
  - “pen” is more similar to “pencil” than it is to “house”!

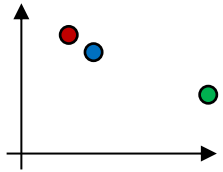
1-hot encoding

```
Pen    → [ 1, 0, 0 ]
Pencil → [ 0, 1, 0 ]
House  → [ 0, 0, 1 ]
```



Word embeddings

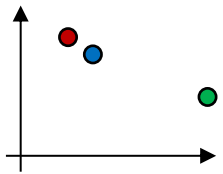
```
Pen    → [ 0.2, 0.9 ]
Pencil → [ 0.3, 0.8 ]
House  → [ 1.2, 0.3 ]
```



From text to vectors

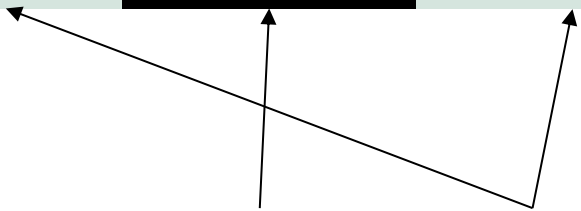
- Many techniques exist (word2vec, FastText, GloVe, etc.)!
- All based on learning to solve "*a problem*" framed on a large training corpora
- Words are projected to a vector space that captures different aspects of each word's meaning
  - The space is *learned* as a byproduct of solving "*the problem*"
  - High-dimensional (300+ dimensions), dense

```
Pen    → [ 0.2, 0.9 ]
Pencil → [ 0.3, 0.8 ]
House  → [ 1.2, 0.3 ]
```



Framing the right *problem*

I used a [redacted] to write the essay



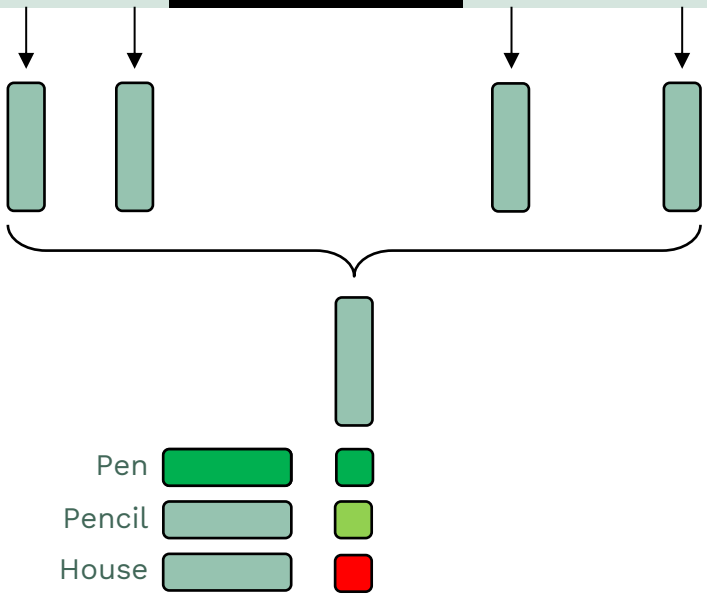
Problem: Can you fill the *blank* , given some *context*?

In other words, we estimate the probability that each word w is the correct one

$$P(x_i = w) = P(x_i = w | x_{i-1}, x_{i-2}, \ldots, x_{i+1}, x_{i+2}, \ldots)$$

Solving the right *problem*

I used a [redacted] to write the essay



Solution:

1. Assign each context word to a vector (random, at first!)
2. Aggregate all vectors (e.g., sum them!)
3. Assign each candidate output word to a vector (random too, at first!)
4. Compute the distance between context and all possible output words (e.g., dot product)
5. Find the word that best matches the context
6. Is it the *correct* word?
7. Adjust the vectors accordingly (via *gradient descent*)

… Rinse and repeat!

- The same process is applied on millions of sentences!
- Similar words are found in similar contexts
- To solve the previous problem, the word vectors of similar words must be similar!

I used a pencil to write the essay

You used my pen to write a letter

…

word2vec

word2vec does all of this, with some caveats:

- Problem type:
  - The previous problem is called CBOW (Continuous Bag Of Words)
  - Alternatively, there is also the skip-gram problem: from the middle word, predict context

- Optimizations (To use a larger vocabulary & more data):
  - Hierarchical softmax: reduce number of predictions used to produce scores of all words
  - Negative sampling: predict if a given word is the correct one (binary problem)

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems 26 (2013).

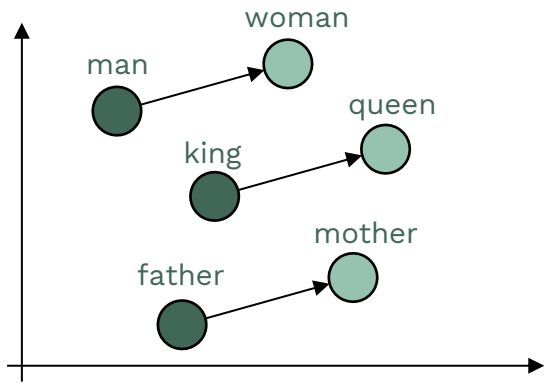
king − man + woman = ?

We can perform operations (e.g., additions, subtractions) between vectors!

An interesting property emerges: some concepts (transformations) are consistent in the space!

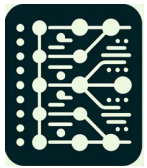
Analogy puzzles like “man is to king as woman is to *what*?” can be addressed as:

$$v_{king} - v_{man} + v_{woman} \approx v_{queen}$$



Sequence-aware models

1d CNN & RNN



1-dimensional Convolutional Neural Networks

Efficient at capturing *local patterns* within fixed-length segments of the sequence

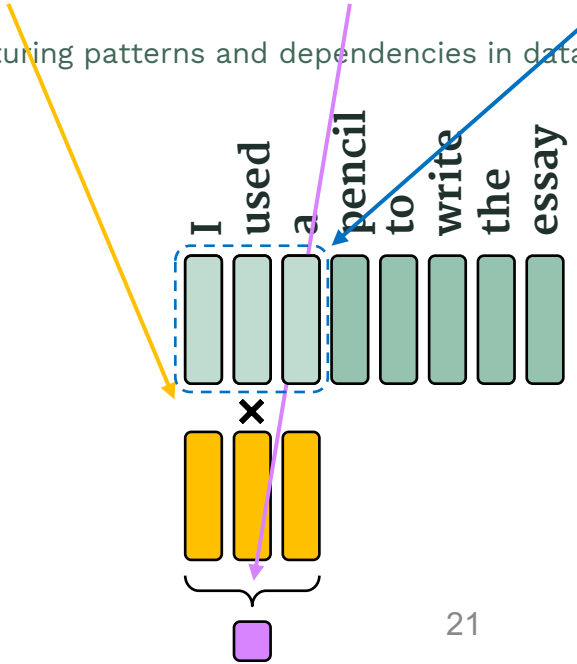
Recurrent Neural Networks

Better at capturing *long-term dependencies* by maintaining memory across the entire sequence

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

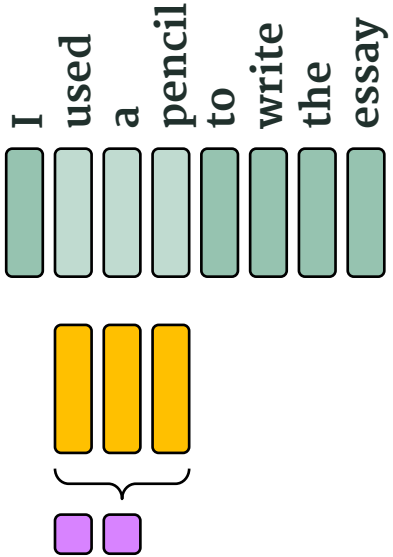
1-dimensional CNNs

- Designed for sequential data like time series, audio signals, and text.
- Uses *convolutional filters* to extract *features* from *local segments* of the input sequence.
- Efficient at capturing patterns and dependencies in data with spatial or temporal proximity.



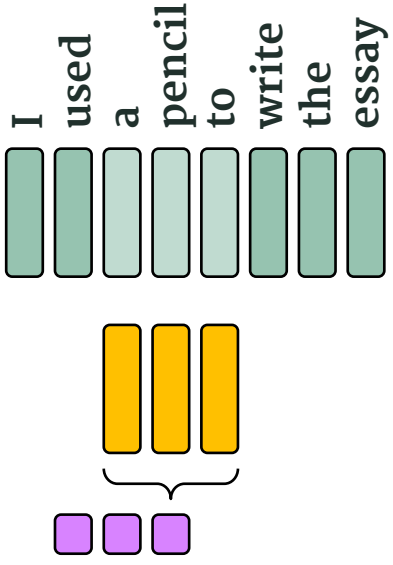
1-dimensional CNNs

- Designed for sequential data like time series, audio signals, and text.
- Uses *convolutional filters* to extract *features* from *local segments* of the input sequence.
- Efficient at capturing patterns and dependencies in data with spatial or temporal proximity.



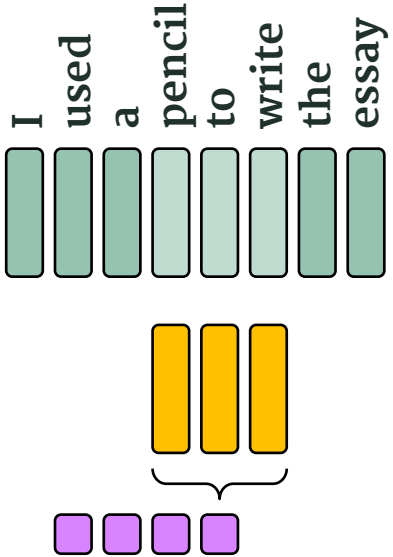
1-dimensional CNNs

- Designed for sequential data like time series, audio signals, and text.
- Uses *convolutional filters* to extract *features* from *local segments* of the input sequence.
- Efficient at capturing patterns and dependencies in data with spatial or temporal proximity.



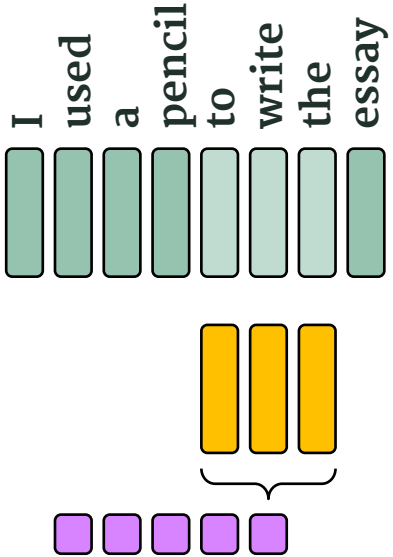
1-dimensional CNNs

- Designed for sequential data like time series, audio signals, and text.
- Uses *convolutional filters* to extract *features* from *local segments* of the input sequence.
- Efficient at capturing patterns and dependencies in data with spatial or temporal proximity.



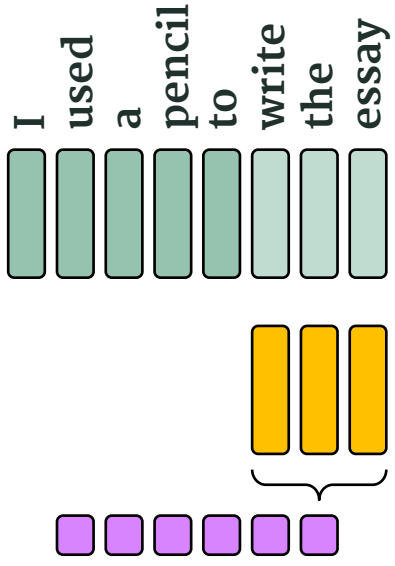
1-dimensional CNNs

- Designed for sequential data like time series, audio signals, and text.
- Uses *convolutional filters* to extract *features* from *local segments* of the input sequence.
- Efficient at capturing patterns and dependencies in data with spatial or temporal proximity.



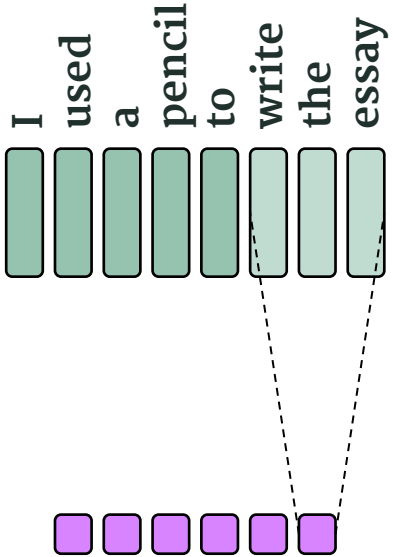
1-dimensional CNNs

- Designed for sequential data like time series, audio signals, and text.
- Uses *convolutional filters* to extract *features* from *local segments* of the input sequence.
- Efficient at capturing patterns and dependencies in data with spatial or temporal proximity.



1-dimensional CNNs

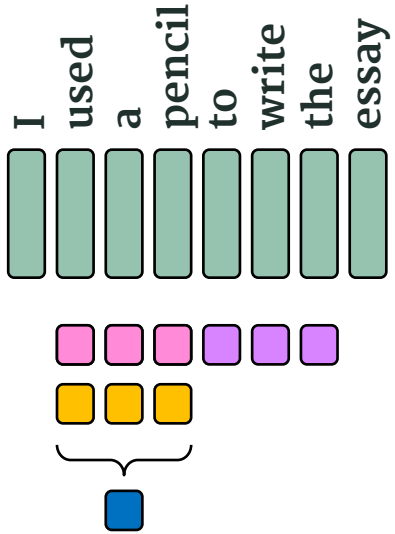
- Designed for sequential data like time series, audio signals, and text.
- Uses *convolutional filters* to extract *features* from *local segments* of the input sequence.
- Efficient at capturing patterns and dependencies in data with spatial or temporal proximity.



Each new features encodes some context!

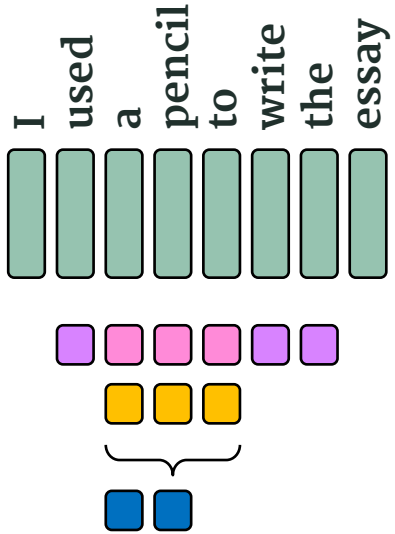
1-dimensional CNNs

- Designed for sequential data like time series, audio signals, and text.
- Uses *convolutional filters* to extract *features* from *local segments* of the input sequence.
- Efficient at capturing patterns and dependencies in data with spatial or temporal proximity.



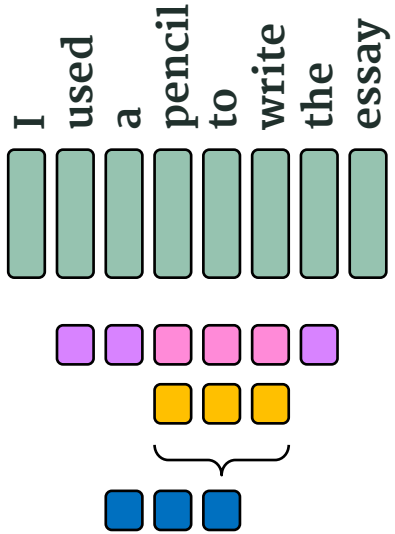
1-dimensional CNNs

- Designed for sequential data like time series, audio signals, and text.
- Uses *convolutional filters* to extract *features* from *local segments* of the input sequence.
- Efficient at capturing patterns and dependencies in data with spatial or temporal proximity.



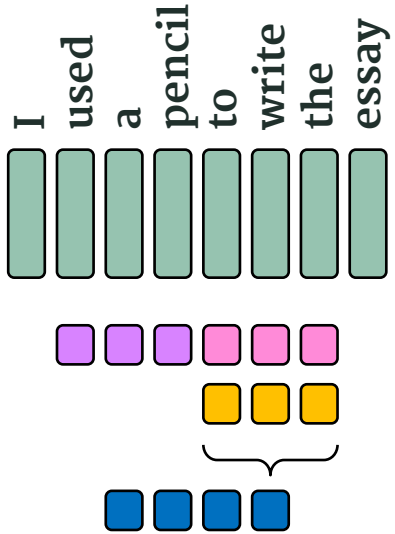
1-dimensional CNNs

- Designed for sequential data like time series, audio signals, and text.
- Uses *convolutional filters* to extract *features* from *local segments* of the input sequence.
- Efficient at capturing patterns and dependencies in data with spatial or temporal proximity.



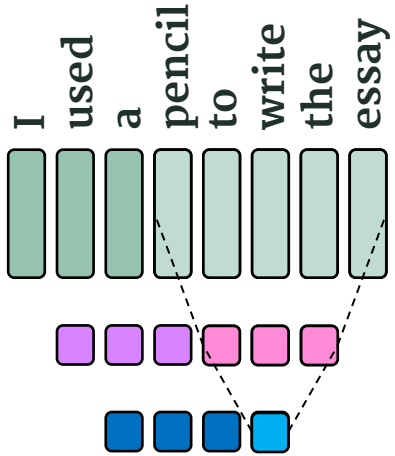
1-dimensional CNNs

- Designed for sequential data like time series, audio signals, and text.
- Uses *convolutional filters* to extract *features* from *local segments* of the input sequence.
- Efficient at capturing patterns and dependencies in data with spatial or temporal proximity.



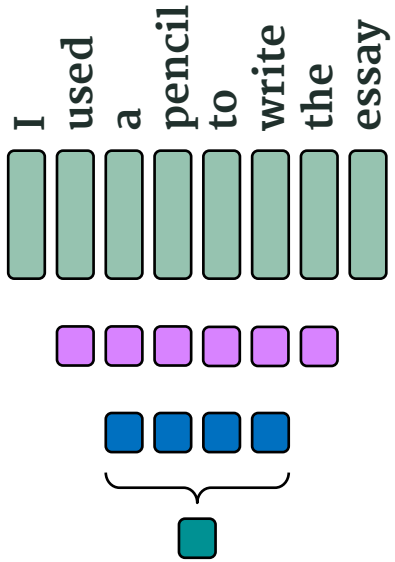
1-dimensional CNNs

- Designed for sequential data like time series, audio signals, and text.
- Uses *convolutional filters* to extract *features* from *local segments* of the input sequence.
- Efficient at capturing patterns and dependencies in data with spatial or temporal proximity.



1-dimensional CNNs

- Designed for sequential data like time series, audio signals, and text.
- Uses *convolutional filters* to extract *features* from *local segments* of the input sequence.
- Efficient at capturing patterns and dependencies in data with spatial or temporal proximity.



Pros & cons of 1d CNN

😄

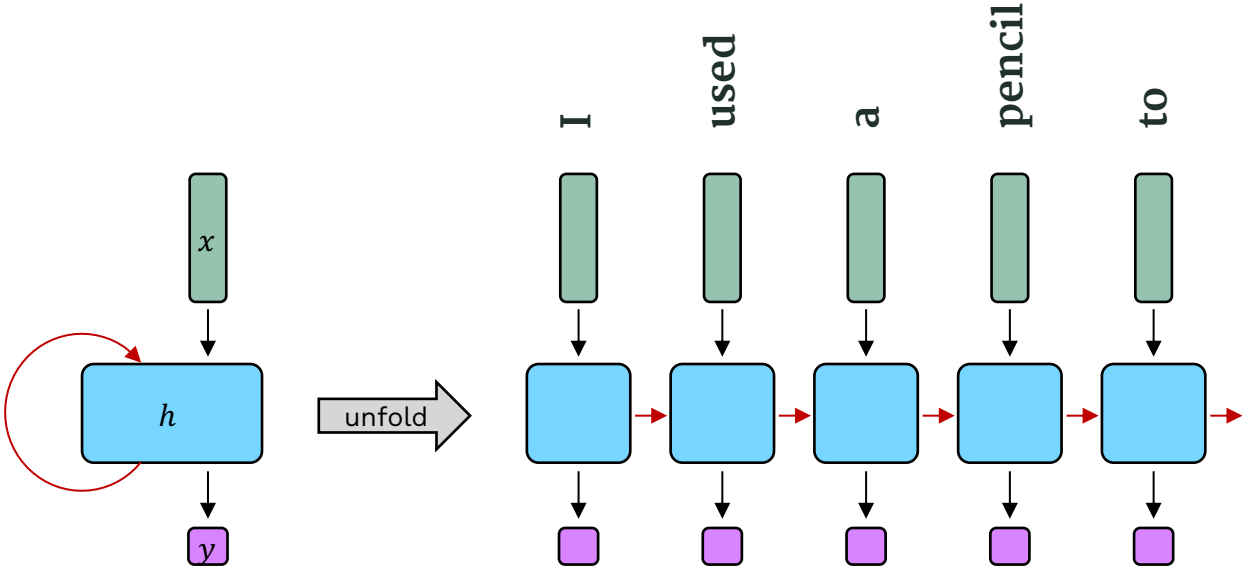
- **Translation Invariance**
  - Recognizes patterns despite timing shifts.
- **Less prone to overfitting**
  - Shared weights and pooling layers enhance generalization.
- **Automated feature extraction**
  - Efficiently learns key features with minimal preprocessing.
- **Parallel Processing**
  - Speeds up training/inference by processing data simultaneously.

😥

- **Limited Context**
  - Struggles with long-term dependencies due to fixed kernel size.
- **Sequence Generation**
  - Not ideal for tasks requiring output dependent on previous data.
- **Complex Design**
  - Requires careful tuning of kernel sizes and filter numbers.

Recurrent Neural Networks

- Designed for sequential data like time series, audio signals, and text.
- Can remember past inputs through *internal state*
- Best where context from previous data points influences current and future data



Pros & cons of RNN

😄

- **Specifically for Sequential Data**
  - Easily processes varying sequence lengths for complex tasks
- **Temporal Dependencies**
  - Can model long-term data relationships via recurrent connections.
- **Effective Memory Use**
  - Uses hidden states to retain relevant past information.

😥

- **Vanishing/exploding gradients**
  - Training with long sequences may be problematic!
- **Sequential Processing**
  - Sequential architecture slows down training: no parallel processing efficiency.
- **Memory Limitations**
  - Struggles with long sequences due to heavy reliance on hidden state memory.

Banking data applications

Modeling TXN Patterns

Each customer (account) produces a sequence of transactions

The sequential nature of transactions can be modeled with sequential ML techniques

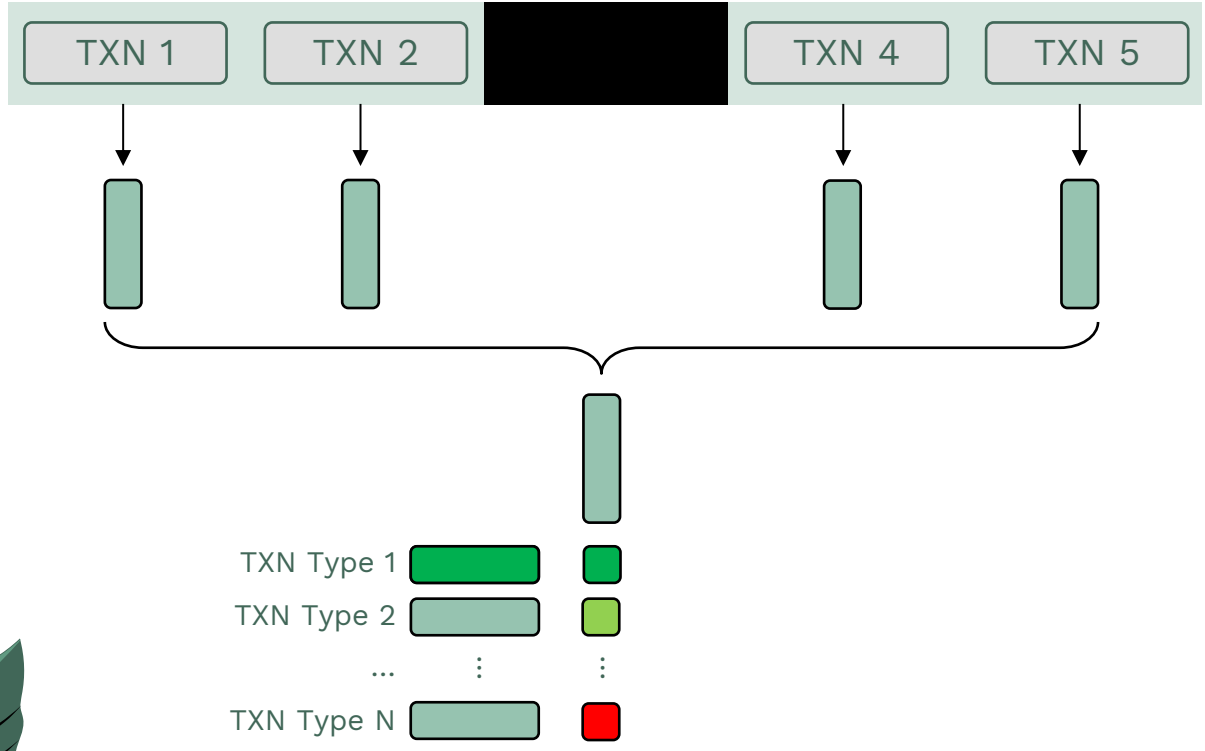
We can model each transaction as:

- Transaction type (wire transfer, withdrawal, deposit, etc)
- Amount
- Type of operation (credit/debit)
- Currency
- Date & time
- Counterpart (for some transactions)
- ...

TXN

Through discretization (amounts, dates, times) we can define a *vocabulary* of transactions

Vectorizing transactions



What can we expect?

- The ”context” of a transaction represents the transactions done before/after the target one
- With word2vec, we can extract semantic similarities between transactions
- These vector representations can be used, for instance, for:
  - *Semantic analysis of transactions/customers*
  - *Unexpectedness of transactions*
  - *Representations for downstream tasks*

Semantic analysis

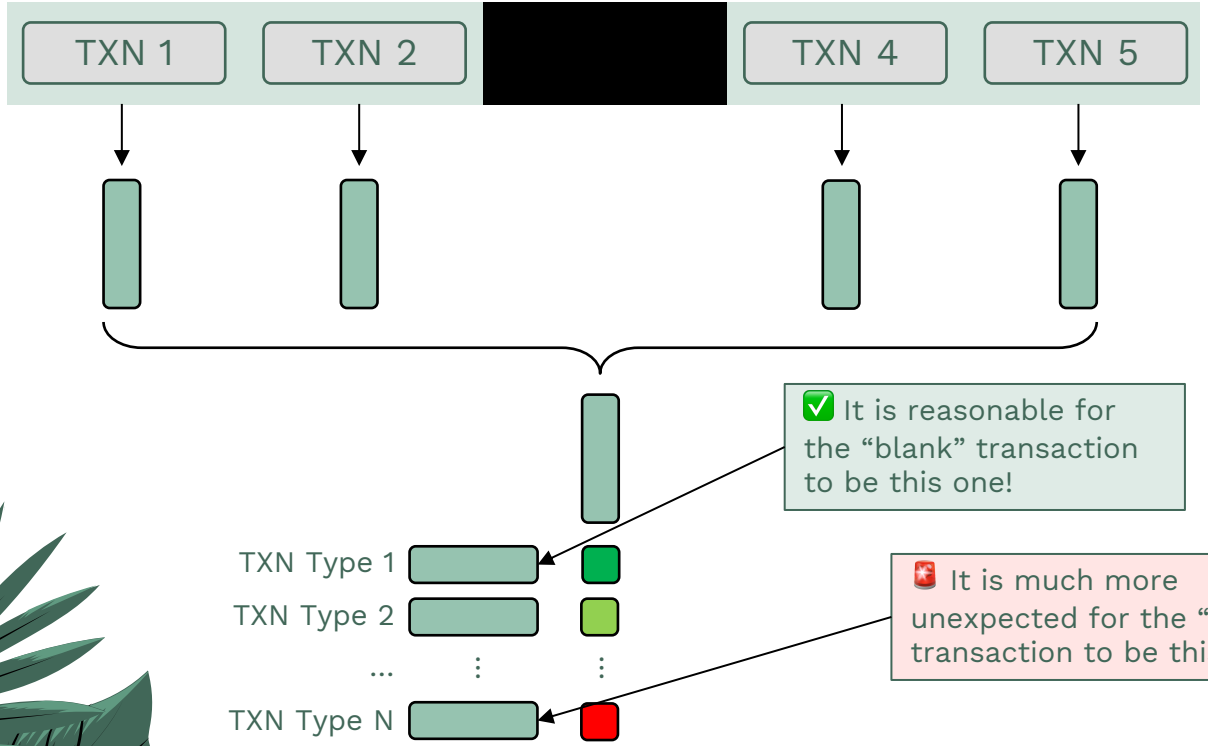
By considering the vector representations of transactions, we can address questions like:

- “To which other *transactions* is this *transaction* most similar to?”
- “Which clusters of *transactions* can we identify?”

We can represent each *customer* as the (vectorial) sum of its *transactions*!

- “To which other *customers* is this *customer* most similar to?”
- “Which clusters of *customers* can we identify?”

Unexpectedness of TXNs



Downstream tasks

Vector representations (+ other sequence-based techniques) can be used for:

- Supervised tasks (e.g., via 1d CNN, RNN)
  - Classification
    - Credit approval, churn prediction
  - Regression
    - Credit scoring, risk management
- Unsupervised tasks
  - Clustering
    - Customer segmentation
  - Anomaly detection
    - Fraud detection, Anti-money laundering monitoring

Recap & conclusions

Various approaches in literature help us approach sequential data

- *Word embeddings* help us encode each point based on their context
- *1d CNNs* can extract local patterns of interest
- *RNNs* can model possibly long sequences by preserving a state (“memory”)

Recent *transformer-based* approaches have achieved state of the art performance

- Their training, however, requires significant amounts of data and resources

In a banking setting, the above approaches *may* be helpful!

- However, experimental results are needed to bolster that claim!

Thank you :)

Flavio Giobergia

✉ flavio.giobergia@polito.it
𝕏 @fgiobergia