Deep Associative Semantic Neural Graphs
for Knowledge Representation and Fast Data Exploration

AGH University of Science and Technology
Adrian Horzyk
horzyk@agh.edu.pl
Google: Horzyk
Brain-Like Associative Processes can be used to organize and associate data in deep neural structures...
Objectives and Contribution

- Implementation of associative mechanisms inspired by brains.
- Construction of deep associative semantic neural graphs DASNG for associative representation of the data stored in relational databases.
- Introduction of a new associative spiking model of neurons that can quickly point out related data and entities and be used for inference.
- Innovation in data storage, organization, access, and management that combines, integrates, aggregates & associates various data collections.
- Implementation of a new mechanism of data access and data processing.
- Efficient representation of wider range of data relations directly in the structure, especially horizontal and vertical relations between entities.
- Replacement of time-consuming procedures by the associative structure which significantly reduces the computational complexity of various operations on data and entities, especially of the search operations.
Limitations of contemporary computers

Contemporary computers:

- are limited by the limitations of the Turing machine computational model,
- use array RAM hindering the implementations of neural graphs,
- separate the data from the program and the memory from the CPU or GPU,
- execute instructions sequentially,
- use synchronous parallelism in the GPUs which does not go hand in hand with the way the neurons work in brains.

Such an environment is not beneficial for simulating asynchronous neurons in brain-like graph structures which use a time approach!
Brains and Neurons

- execute stimulations parallel and often asynchronously,
- automatically, fast and context-sensitively associate data and entities,
- use a complex graph memory structure and parallel procedures,
- integrate the memory with the program which use previous knowledge,
- use time approach for temporal and contextual computations,
- are not limited by the Turing machine computational model.
Selected Drawbacks of Relational Data Model

- The lack of representation of vertical relations between objects in each table.
- The necessity to find out vertical relations between objects as order, similarity...
- The more entities are stored in the table the bigger problem we have (BIG DATA).
- Non-efficient representation of the duplicated data in the same or various tables.
- Non-associated parameters and data in various tables describing the same categories.
1. We have to use various search routines (inside SELECT) to retrieve information.
2. Each search routine costs time and power because they use many nested loops.
3. The results of the work are often unsaved or non-suitable for further operations.
4. We must store many entities of duplicates which are not aggregated in this model.
5. We must use indices because parameters are not ordered or related due to their values.
Associative Transformation of Relational Database

Small hospital database
In order to **accelerate search routines**, we should **associate all related data and entities (objects)** represented in the database, namely:

- Sort all orderable attribute values,
- Directly connect related objects by link-tables,
- Aggregate all the same values (duplicates) of the same categories.

In consequence, related objects will be **quickly available** and will not require to be searched, indexed, or compared in many nested loops.

**All duplicated values** of the same category occurring in the same or different tables will be **aggregated and sorter**.

In result, we achieve **an associative graph structure** representing all **horizontal** and additionally **vertical relations** between data and objects.

**The graph nodes** contain the **numbers of aggregated duplicates**.

**The graph connections** contain the **information about the strength of relations** of the connected objects or values.
Associative Graph Structure Can Replace Many Search Operations

AVB-TREES = FAST ACCESS

SENSORS

SENSEThe fields containing sensors representing attribute values

ORDERED

AGGREGATION
OF DUPLICATES

DASNG

NEURONS
DASNG – Deep Associative Semantic Neural Graphs

- **D – deep** – means the ability to represent various data relations in the specific deep neural network structure.

- **A – associative** – stands for the way human brain works, it allows for the fast availability of various data accordingly to the context in which these data occurred in the data set used to develop DASNG neural network.

- **S – semantic** – means that all semantically related objects are directly or indirectly connected to enable fast access to them if necessary.

- **N – neural** – because a special associative model of spiking neurons is used to represent attribute data, their ranges or subsets, as well as objects, clusters, classes etc.

- **G – graph** – because all neurons are connected in a sparse graph structure that represents associations between data and objects.
DASNG Features

- **DASNG** contains all horizontal relations between objects that are implemented in relational model.
- **DASNG** naturally implements many vertical relations between objects thanks to aggregations of duplicates and connections between neurons representing similar (ordered) attribute values.
- **DASNG** always puts new data into the context of other stored data.
- **DASNG** use an associative spiking neurons to implement reactive functionality and automatic inference according to the initial context.
- **DASNG** significantly decreases computational complexity for many operations because it replaces complex operations by its structure.
- **DASNG** replaces many time consuming loops on tabular structures.

**DEF:** We say that the structure replaces operations performed on another data structure when the computational complexity of the operations on that structure decreases to constant computational complexity $O(1)$. 
DASNG construction for DB
Transforms only these tables for which all foreign keys are already represented by the neurons in the DASNG:

Possible sequence of transformation of tables
DASNG construction for DB

The table NURSES is added to the empty DASNG network.
DASNG construction for DB

The table DOCTORS is added to the DASNG network.
DASNG construction for DB

The table ROOMS is added to the DASNG network.
DASNG construction for DB

The table NURSEROOM is added to the DASNG network.
DASNG construction for DB

The table PATIENTS is added to the DASNG network.
The table DOCTORPATIENT is added to the DASNG network.
Result of the associative transformation of the DB to the DASNG network:

No duplicates and all values are sorted and quickly accessible!
DASNG uses AVB-trees for fast attribute data access

AVB-tree is a new self-ordering and self-balancing tree structure that enables to efficiently organize attribute values and achieve very fast access to all stored feature values and objects in the DASNG network.

AVB-trees are very similar to B-trees but AVB-trees additionally aggregate and count up all duplicated values.

The aggregations of duplicates result in typically much smaller number of nodes of AVB-trees than achieved for B-trees for the same collection of data.

Search operations are usually also faster taking usually constant time!

THE COMPARISON OF AVB-TREE AND B-TREE

counters of duplicates
**AVB-trees construction**

**AVB-trees** are constructed similarly to B-trees but duplicates are aggregated (represented once) and counted up:
AVB-trees are fully self-balancing:

When the **node** contains more than two keys (values or sensors) it is automatically divided as shown in this sample presenting the intermediate operations on this AVB-tree:
AVB-trees construction algorithm: INSERT operation

1. Start **from the root** and **go recursively down** along the edges to the descendants **until the leaf** is not achieved after the following rules:

   - if one of the keys stored in the node **equals** to the inserted key, **increment** the counter of this key, and finish this operation,
   - else **go to the left child** node if the inserted key is **less than** the leftmost key in the node,
   - else **go to the right child** node if the inserted key is **greater than** the rightmost key in the node,
   - else **go to the middle child** node.
2. When the leaf is achieved:

- and if the inserted key is equal to one of the keys in this leaf, increment the counter of this key, and finish this operation,

- else insert the inserted key to the keys stored in this leaf in the increasing order, initialize its counter to one, and go to step 3.
3. If the number of all keys stored in this leaf is greater than two, divide this leaf into two leaves in the following way:

- let the divided leaf represent the leftmost (least) key together with its counter;
- create a new leaf and let it to represent the rightmost (greatest) key together with its counter;
- and the middle key together with its counter and the pointer to the new leaf representing the rightmost key pass to the parent node if it exists, and go to step 4;
- if the parent node does not exist, create it (a new root of the AVB-tree) and let it represent this middle key together with its counter, and create new edges to the divided leaf representing the leftmost key and to the leaf pointed by the passed pointer to the new leaf representing the rightmost key. Next, finish this operation.
4. **Insert** the passed key together with its counter to the key(s) stored in this node in the increasing order after the following rules:

- if the key comes from the left branch, insert it on the left side of the key(s);
- if the key comes from the right branch, insert it on the right side of the key(s);
- if the key comes from the middle branch, insert it between the existing keys.

5. **Create a new edge** to the new leaf or node pointed by the passed pointer and **insert this pointer** to the child list of pointers immediately after the pointer representing the edge to the divided leaf or node.
6. If the number of all keys stored in this node is greater than two, divide this node into two nodes in the following way:

- let the existing node represent the leftmost (least) key together with its counter;
- create a new node and let it represent the rightmost (greatest) key together with its counter;
- the middle key together with its counter and the pointer to the new node representing the rightmost key pass to the parent node if it exists and go back to step 4;
- if the parent node does not exist, create it (a new root of the AVB tree), let it represent this middle key together with its counter, and create new edges to the divided node representing the leftmost key and to the node pointed by the passed pointer to the new node representing the rightmost key. Next, finish this operation.
DASNG uses Sensors and Associative Spiking Neurons

Sparsely and contextually connected neural networks play an important role in the associative processes in the brain where knowledge is represented. The DASNG uses models of neurons which incorporate the concept of time.
The fundamental problem is to propose the model that explains **how information is encoded and decoded** by a series of pulses, i.e. action potentials?!

The **fundamental question of neuroscience** is to determine **whether neurons communicate by a rate or temporal code**?

Temporal coding suggests that a single spiking neuron can replace hundreds of hidden units on a sigmoidal neural network. Is that true?

Experiments on DASNG networks revealed that both **time and rate** have an appropriate influence on postsynaptic neuron activity, and thus on what this neuron represents.
How do Associative Spiking Neurons work and Influence other Neurons?

Each activation of the neuron $O_j$ stimulates and activates the neuron $R_i$, which stimulates the neighboring sensory neurons $R_{i+1}$ and $R_{i-1}$ with the force equal to the weights of these connections, i.e. 0.8 and 0.6, appropriately. It is therefore necessary to stimulate these neurons twice, so that, with regards to relaxation, they achieve a total stimulus greater than their activation thresholds $\theta = 1$. This will allow them for activation and then to start stimulation of the connected neurons, e.g. the neuron $O_k$. 
How do Associative Spiking Neurons work and Influence other Neurons?

As we can notice, the neuron $O_k$ needs to be stimulated triple times through the connection coming from the neuron $R_{i+1}$ and weighted with 0.5 to reach the activation threshold $\theta = 1.0$.

When a neuron is not externally stimulated, the relaxation and refraction processes try to restore the resting state in it.
How do Associative Spiking Neurons work and Influence other Neurons?

The sensory neurons $R_{i+1}$ and $R_{i-1}$ are stimulated with different strength according to the weights (0.8 and 0.6) of connections coming from the neuron $R_i$. It induces different excitation levels inside them and different activation moments. The neuron $R_{i+1}$ achieves this threshold earlier than the neuron $R_{i-1}$, so the neuron $R_{i+1}$ starts earlier to stimulate the neuron $O_k$ than the neuron $R_{i-1}$ starts to influence the neuron $O_m$. Thus, the neuron $O_k$ will be activated earlier than the neuron $O_m$. It implies greater similarity of the object represented by the neuron $O_k$ than by the neuron $O_m$. This is consistent with intuition of real similarity.
How do Associative Spiking Neurons work and Influence other Neurons?

The small shift in activation of the neurons $O_k$ and $O_m$ may seem to be insignificant or negligible, but this phenomenon is crucial for the working way of biological neural networks as well as of the introduced associative neural graphs DASNG.

The difference in activation time of these neurons representing different objects informs us of weaker or stronger associations with these objects, i.e. less or greater similarity of them.

In this way, associative spiking neurons automatically conclude, revealing their various relationships with other objects and data represented by other connected neurons.
Orderable sensory neurons are connected, the connections are weighed expressing similarity:

\[ W_{R_{vi}, R_{vj}}^{a_k} = 1 - \left| v_i^{a_k} - v_j^{a_k} \right| \]

The connections between the sensory and object neurons are weighted in the following way:

\[ W_{R_{vi}, O_j^T}^{a_k} = \frac{1}{\| v_i^{a_k} \|} \quad W_{O_j^T, R_{vi}}^{a_k} = \theta R_{vi}^{a_k} = 1 \]

The weights of synaptic connections between various object neurons are computed on the basis of the number of objects represented by the object neurons of the considered layer of the DASNG, which represents a single database table. If the given object neuron of the considered layer is connected to \( M \) object neurons of another layer, then the weight is computed in the following way:

\[ W_{O_j^T, O_k^T} = \frac{1}{N_{j,T_{n}}^{T_{n}}} \approx \frac{1}{M} \quad W_{O_k^T, O_j^T} = \frac{1}{N_{k,T_{n}}^{T_{n}}} \approx \frac{1}{N} \]

where \( N_{k,T_{n}}^{T_{n}} = N = 1 \) for the relations one-to-many (1:M) and the relations many-to-many (N:M). The equation is precise when there are no duplicates of the whole records in the database. We need to create separate lists of connections in each neuron to represent connections to neurons of various layers in order to easily compute the number of objects \( N_{j,T_{m}}^{T_{n}} \) or the number of connections \( M \).
Activation Thresholds

Activation thresholds of sensory neurons:

\[ \theta_{R_{v_i}^a} = 1 \]

Activation thresholds of object neurons:

\[
\theta_{O_j^{T_n}} = \begin{cases} 
1 & \text{if } \sum_{R_{v_i}^a} w_{R_{v_i}^a, O_j^{T_n}} \geq 1 \\
\sum_{R_{v_i}^a} w_{R_{v_i}^a, O_j^{T_n}} & \text{if } \sum_{R_{v_i}^a} w_{R_{v_i}^a, O_j^{T_n}} < 1 
\end{cases}
\]

The above definition of the activation threshold allows for activation of an object neuron whenever it is stimulated by the \textbf{whole defining combination} of this neuron, or when it is stimulated by a \textbf{sufficiently representative subset of rare or unique features} defining this neuron, e.g. if a feature defines only one object neuron, then it is enough to recognize it when this feature appears.
Linear Approximation of the Internal Neuronal Processes

The DASNG associative spiking neurons (ASNs) uses a **linear approximation** of all processes. This greatly simplifies and speeds up calculations of neuronal states:

Each neuron creates an **internal neuronal process queue (IPQ)** of successive processes ordered after the time of their beginning. New processes are added to this queue on the basis of stimuli coming from other neurons or a sensor.

In order to appropriately order parallel processes of all neurons in the DASNG in time, there is used a **global event queue (GEQ)** and each event watches a single process.
The neuronal internal process queue (IPQ) combines external stimuli with internal processes and chronologically orders neuronal processes to not overlap in time.
Combining the Internal Processes with a New Stimulus

\[
\hat{P}_0 = (\hat{r}_0, t_s, d_0 - (t_s - t_0), s_0 \cdot \frac{d_0 - (t_s - t_0)}{d_0} + s_s \cdot \frac{d_0 - (t_s - t_0)}{d_s}, \hat{p}_0) \\
\hat{P}_1 = (\hat{r}_1, t_0 + d_0, d_s - (d_0 - (t_s - t_0)), s_s \cdot \frac{d_s - (d_0 - (t_s - t_0))}{d_s}, \hat{p}_1) 
\]
Inference using the DASNG network

Neuronal inference can be achieved by stimulation of sensors!
Inference using the DASNG network

We can quickly get any associated information waiting for neuronal activity!
Inference using the DASNG network

Connections representing associations allow for further inferences.
Inference using the DASNG network

Indirectly associated information is also available after short time!
DEEP ASSOCIATIVE SEMANTIC NEURAL GRAPHS (DASNG) can be used to:

- transform databases into the reactive associative data structure,
- create deep neural network architectures for spiking neurons,
- represent complex objects contextually alike in databases, additionally specifying the strength of associated (related) objects (entities),
- filter values or objects (entities) according to the initial stimulation(s),
- quickly return objects sorted after any combination of attributes,
- immediately get minima and maxima of any attribute,
- inference on the basis of the initial context used for stimulation of the DASNG network using sensors and sensory neurons,
- create knowledge-based cognitive and artificial intelligence systems.
APPLICATIONS

DEEP ASSOCIATIVE SEMANTIC NEURAL GRAPHS (DASNG) are planned to be used in CERN in A Large Ion Collider Experiment (ALICE) in O2 and O3 run for quality control and Big Data analysis in real-time.

DASNG can be parallelized and draw conclusions in constant time.


9. Horzyk, A., How Does Generalization and Creativity Come into Being in Neural Associative Systems and How Does It Form Human-Like Knowledge?, Neurocomputing, 2014.