Hasso Plattner Institut

Digital Engineering · Universität Potsdam

```
<!DOCTYPE html>
<html>
 <head>
  .
 </head>
 <body>
  <table>
   <tr>
    <td>..</td>
    <td>..</td>
   </tr>
  </table>
 </body>
</html>
```

PWT: Introduction to Machine Learning

Prof. Dr. Felix Naumann , Hazar Harmouch and Leon Bornemann

SoS-2019

Agenda

- ❑ Group Allocation and Organisational Information
- ❑ Introduction to Machine Learning
- ❑ Overview of parallel and distributed Computation Methods
- ❑ How to read a research paper

Agenda

- ❑ Group Allocation and Organisational Information
- ❑ Introduction to Machine Learning
- ❑ Overview of parallel and distributed Computation Methods
- ❑ How to read a research paper

What is Next?

Hasso Plattner Institut

11.4.2019	Introduction
18.4.2019	Group allocation+mini talk about research
25.4.2019	Search for literature and giving talks
2.5.2019	**Task presentations (summary and contributions)**
9.5.2019	Progress report
16.5.2019	**Technical talk about specific paper (15%)**
23.5.2019	Research Process – practical hints and clues
30.5.2019	**Christi Himmelfahrt**
6.6.2019	**Intermediate presentation (implementation)**
13.6.2019 -27.6.2019	Progress report
4.7.2019	**Present improvements**
11.7.2019-18.7.2019	Progress report
25.7.2019	**Final presentations (20%)**

(10%) Active participation in meetings and discussions

(25%) Quality of implementation and coding style

(30%) Final paper-style submission

Your Research Topics??

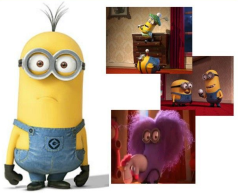
1 Table Header Detection

2 Relational Webtable Detection

3 Table Normalization

Bob

Kevin



Stuart

Basic Related work

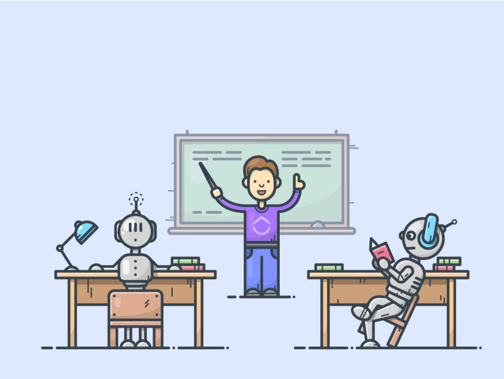
- **Table Header Detection and Relational Webtable Detection**
  - Cafarella, M. J., Halevy, A. Y., Zhang, Y., Wang, D. Z., & Wu, E. (2008, June). Uncovering the Relational Web. In WebDB.
  - Crestan, E., & Pantel, P. (2011, February). Web-scale table census and classification. In Proceedings of the fourth ACM international conference on Web search and data mining (pp. 545-554). ACM.

- **Table Normalization**
  - Braunschweig, K., Thiele, M., & Lehner, W. (2015, October). From web tables to concepts: A semantic normalization approach. In International Conference on Conceptual Modeling (pp. 247-260). Springer, Cham.
  - Wang, D. Z., Dong, X. L., Sarma, A. D., Franklin, M. J., & Halevy, A. Y. (2009, June). Functional Dependency Generation and Applications in Pay-As-You-Go Data Integration Systems. In WebDB.

Agenda

- ❑ Group Allocation and Organisational Information
- ❑ Introduction to Machine Learning
- ❑ Overview of parallel and distributed Computation Methods
- ❑ How to read a research paper

Introduction to Machine Learning

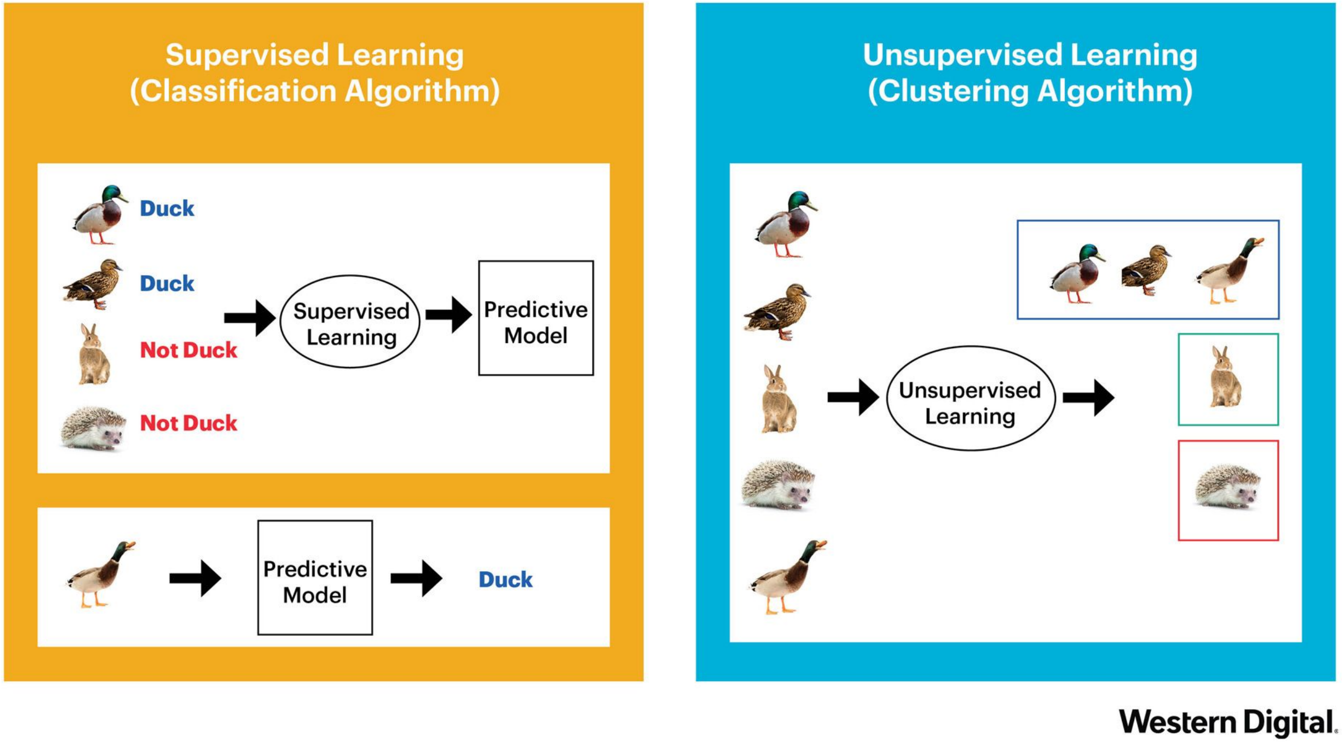


- Supervised vs Unsupervised
  - Supervised Learning Algorithms require Ground Truth
  - Unsupervised Learning Algorithms require “just” data
- Supervised Learning
  - Use Data to learn correct behavior
  - We know what we want but not how
    - Maybe the problem is too complex (has too many nuances)
    - Maybe we are lazy
    - Maybe there is too much data for a human to process

Machine Learning Task Examples

- Unsupervised Learning
  - Clustering Algorithms
    - Given a set of objects, from groups that minimize intra-group distance
  - Anomaly Detection
    - Given a set of objects, find something that does not belong (outliers)
- Supervised Learning
  - Classification
    - Given an object, determine its class (for example E-Mail: Spam/Not Spam)
  - Regression
    - Predict numerical values (for example the price of a stock)

Introduction to Machine Learning



Classification

- Basic Task
  - Given an object and a set of class labels $L = \{L_1,...,L_n\}$ pick the correct label L_{GT}
  - Humans do classification all the time … and are typically pretty good at it
- A **model** is a program that performs the classification
  - Models can be simple
  - If(E-Mail.isAllCaps) return SPAM else return NO-SPAM
- A **Classifier / Classification Algorithm** is able to build models from Training data
  - **Training Data** is a set of objects with their correct/true labels given
- **Feature Based Classification**
  - Objects are represented as a set of attributes (= tuples of a table). The schema of the table is also called the **feature space** (typically if all attributes are numeric)

Feature Based Classification

Electric Pokemon

Bug Pokemon

???

Feature Based Classification

<u>Type</u>	Name	Color	Height	Pokedex-Nr
E	Pikachu	Yellow	3.1	25
E	Pichu	Yellow	2.4	172
B	Scyther	Green	5.7	123
B	Burmy	Green	2.7	412
...	...	...	...	...

???	Wormadam	Green	3.7	413

Selected/Extracted Features

Representations of our objects

Representations of new object with unknown class

Things that can go wrong

- Bias
  - If (height < 5) return E else return B
- Overfitting
  - if(NR is in [25,172,]) return E else return B
- Missing crucial training data

- Missing crucial features
  - For example "*Weak against*" / "*strong against*"

Model is too simple

Model perfectly fits the training data but generalizes poorly

This is also a Bug-Type!

Classifier Evaluation

- On previously unseen Data!
- Accuracy= TP+TN/ (TP+TN+FN+FP)
- Precision/Recall/F1-score
  - Recall = TP / (TP+FN)
  - Precision = TP / (TP+FP)
  - F-score is the harmonic mean of precision and recall:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

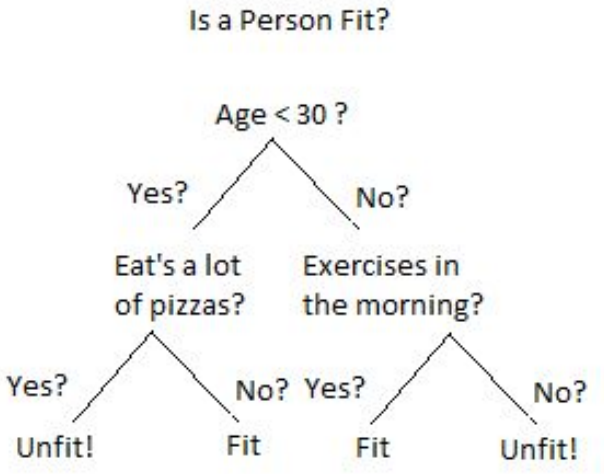
		Actual +	Actual -
Predicted	Y	True positives	False positives
Predicted	N	False negatives	True negatives

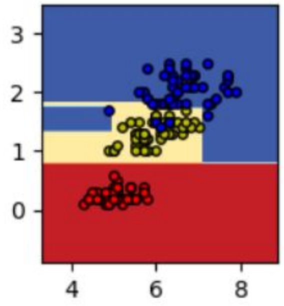
Feature Based Classifiers

- Decision Tree
- Random Forest
- Perceptron
- Support Vector Machines
- Artificial Neural Networks
- ...

Decision Tree

- Idea:
  - Split the training data according to its attributes
  - Goal: Achieve pure leaf-nodes
- Classification Process
  - New, unseen object will land in a leaf node
  - Majority Voting of all Training data in that node
- Split Criterion
  - Entropy /Gini-Index
  - Intuitively: Split on the attribute that results in the purest leaf nodes



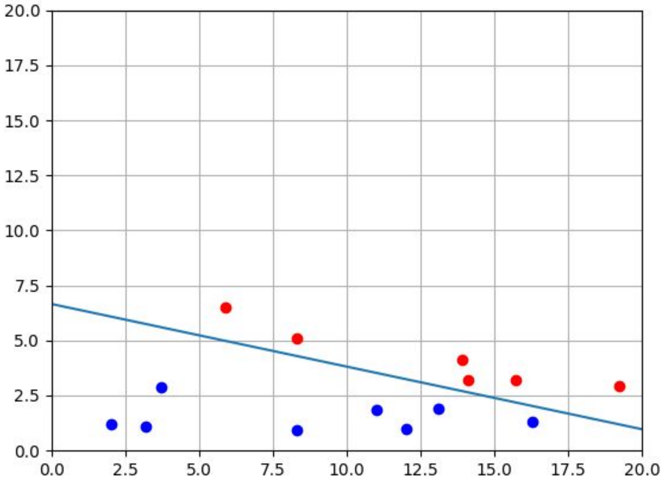


Random Forest

- Idea:
  - Ensemble Learning: Use n different classifiers and aggregate result (majority)
  - If the classifiers are independent and have an accuracy greater 50% the overall accuracy approaches 100% with rising n
  - In this case: build many (~500) decision trees
  - Obtain different trees by subsampling training records or attributes

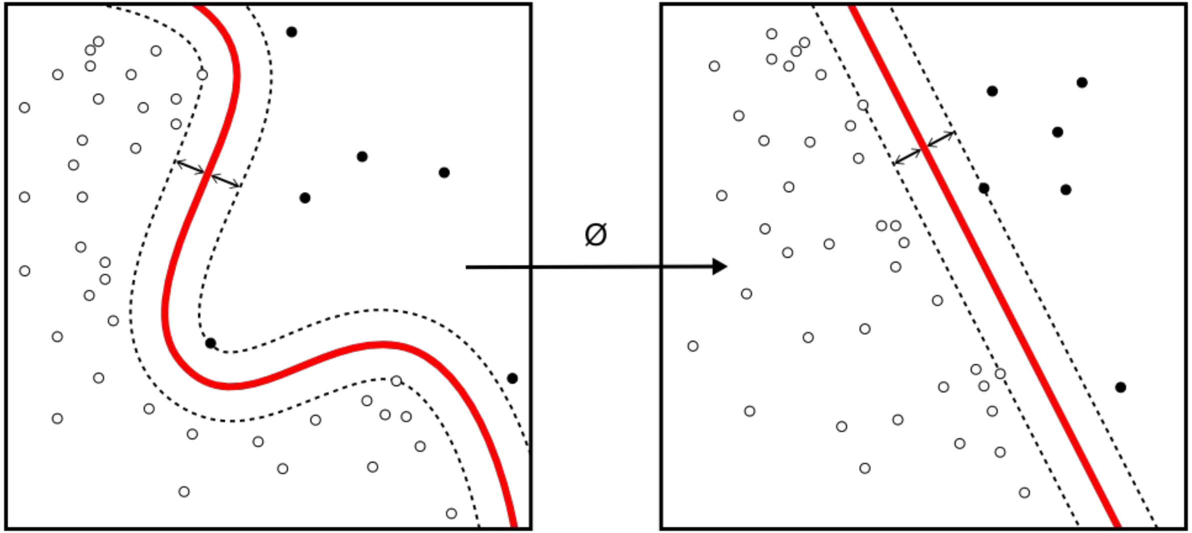
Perceptron

- Perceptron Idea
  - Works on numerical data
  - An object is a point in (Hyper)-Space
  - Separate objects of different classes
    via a line (Hyperplane)
- Training:
  - Initialize random line and iteratively improve it
    By tweaking it’s coordinates
- Classification of new objects:
  - Which side of the Hyperplane are they on?
- Problem: Data might not be separable by linear function



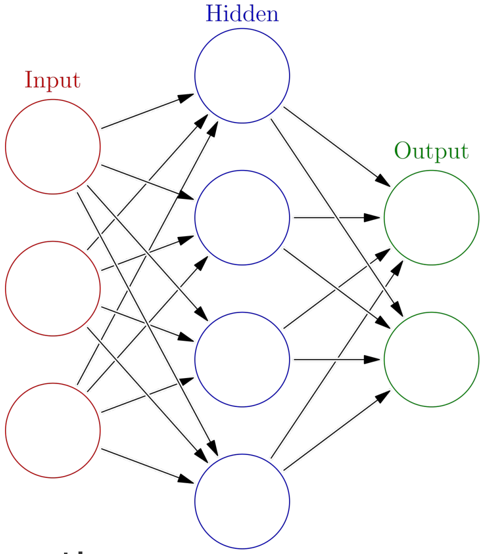
Support Vector Machine

- SVM Idea
  - Same as Perceptron, but use math-magic to map the problem to a different space in which the data is separable by a line



Artificial Neural Network (ANN)

- ANN Idea
  - Use many connected layers of perceptrons, learn their individual weights (importance), use non-linear activation functions
  - Also called Multi-Layer Perceptron (MLP)
- Infinite amount of possible network architectures
  - Certain tasks have “standard” architectures
- “Deep” Learning: Use Many layers, do not pre-filter input
  - State of the art for many research areas
  - Typically requires a lot of training data
- Arguable one of the most difficult classifiers to use (correctly)
- But also (arguably) the most powerful one
- There is a prove that an MLP can approximate any non-linear function



What about Text Features?

- Text can be converted to numerical data
  - Term frequency-inverse document frequency (tf-idf
  - Hashing
  - Word2Vec / Doc2Vec
  - N-Gram Embeddings

Which Classifier to pick?

- Typically matters less than other factors
  - Clean training data
  - A lot of training data
  - The right training data
  - The right features / feature extraction methods
- Apart from that:
  - You should understand the classifier you are using and it should fit your scenario
  - Start with simple ones (for example Random Forest)
  - Move to more complex classifiers only if you have excluded all other possibilities for poor performance (or you want to optimize the last 1-2% of accuracy)

What libraries?

- R and Python both have large machine learning/classification libraries
  - Python’s scikit-learn is probably the most well-known
  - Python’s Tensorflow is often used for ANNs (by Google)
  - Python’s PyTorch (by Facebook)
- Java has WEKA as a library, but it is quite old (no generics oO) and less frequently used
- Links
  - https://pytorch.org/
  - https://scikit-learn.org/stable/
  - https://www.tensorflow.org/
  - www.cs.waikato.ac.nz/ml/weka

Agenda

- Group Allocation and Organisational Information
- Introduction to Machine Learning
- Overview of parallel and distributed Computation Methods
- How to read a research paper

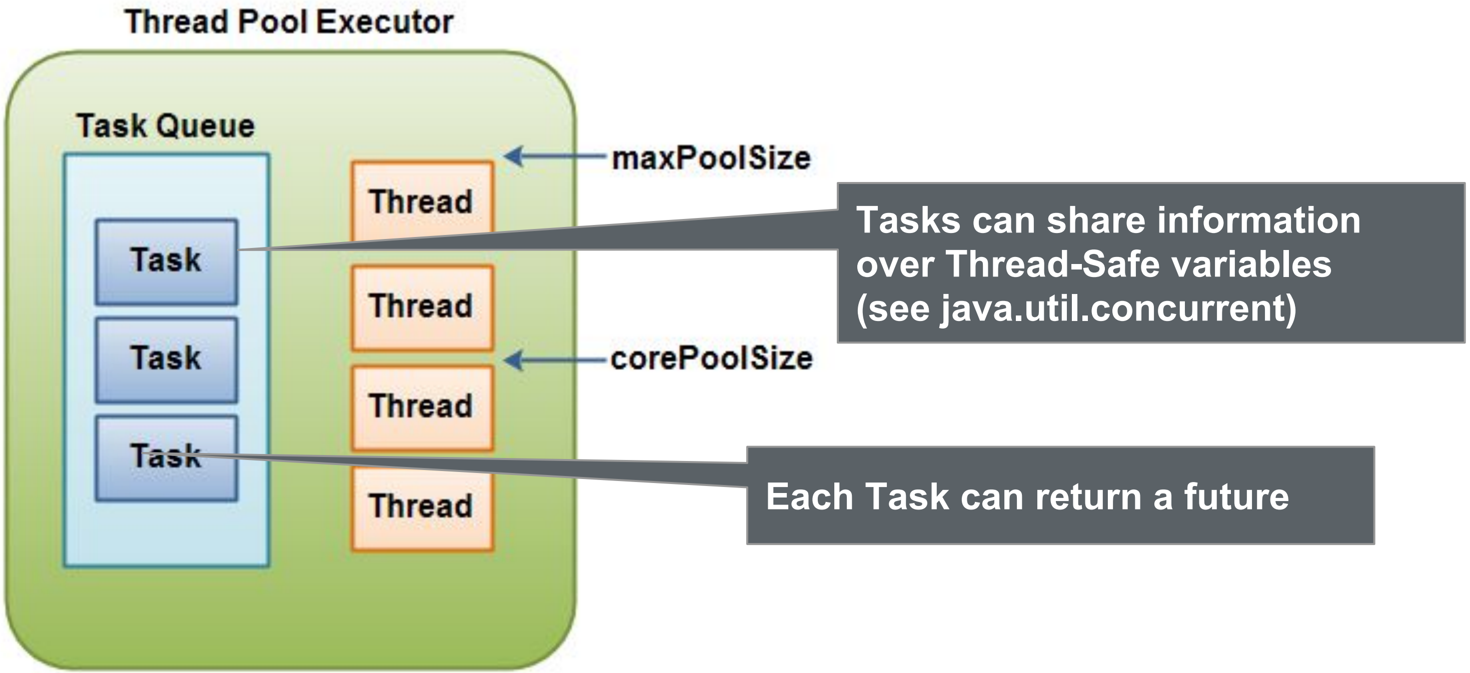
Handling large data

- Scale Up!
  - Execution in parallel (on one machine)
  - Distributed Execution (on many machines

Ways to scale up

- Simply divide the input and start processes in parallel (For example via GNU Parallel)
  - Easy and Surprisingly effective
  - Requires a partitionable program
- Use inner-process parallelism
  - Multiple Threads
  - Potential to use shared Memory / Data Structures
- Use Distributed Applications
  - Distribute Manually
  - Map-Reduce + distributed Filesystem
  - Actor-Programming

Inner Process Parallelism - Java Executors



Inner Process Parallelism - Java Executors

```
public class WorkerThread implements Runnable {

    private String command;

    public WorkerThread(String s){
        this.command=s;
    }

    @Override
    public void run() {
        System.out.println(Thread.currentThread().getName()+" Start. Command = "+command);
        processCommand();
        System.out.println(Thread.currentThread().getName()+" End.");
    }

    private void processCommand() {
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
```

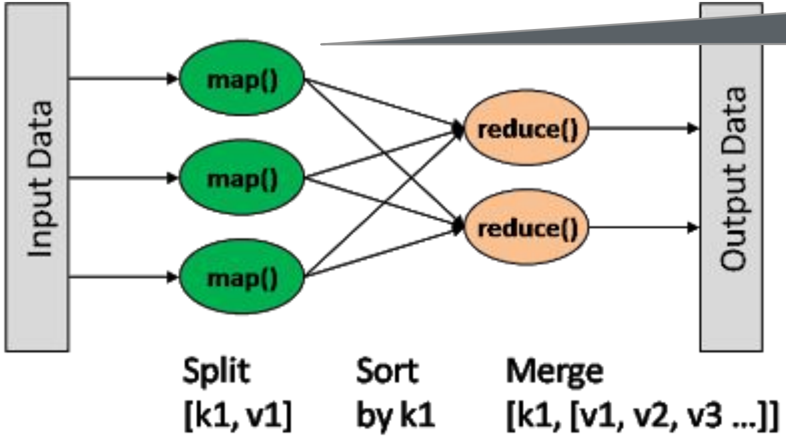
Inner Process Parallelism - Java Executors

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class SimpleThreadPool {

    public static void main(String[] args) {
        ExecutorService executor = Executors.newFixedThreadPool(5);
        for (int i = 0; i < 10; i++) {
            Runnable worker = new WorkerThread("" + i);
            executor.execute(worker);
          }
        executor.shutdown();
        while (!executor.isTerminated()) {
        }
        System.out.println("Finished all threads");
    }
}
```

Bad! - Avoid Busy Waiting!

Distributed Solutions: Map-Reduce

- Idea Map-Reduce: Automatically split processing to different machines
- Map-Reduce: Inspired by functional Programming
  - Map-Function: transforms data records (1 to 1)
  - Reduce-Function: groups and aggregates by group



Data can already be distributed

Distributed Solutions: Apache-Spark

- Implementation of Map-Reduce
- Available for Scala, Java, Python and R
- Scala is the best fit

```scala
val textFile = sc.textFile("hdfs://...")
val counts = textFile.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

Agenda

- Group Allocation and Organisational Information
- Introduction to Machine Learning
- Overview of parallel and distributed Computation Methods
- How to read a research paper

How to read a research article?

- ❑ Big picture:
  - ❑ Title, key words, abstract and conclusion.
  - ❑ Read heading and subheadings.
  - ❑ Check publication date and venue.
  - ❑ Is it relevante?
- ❑ Re-read:
  - ❑ How they model the problem?
  - ❑ How they solve the problem?
  - ❑ How good the results?
- ❑ Summarize
  - ❑ Highlight key points
  - ❑ Take notes

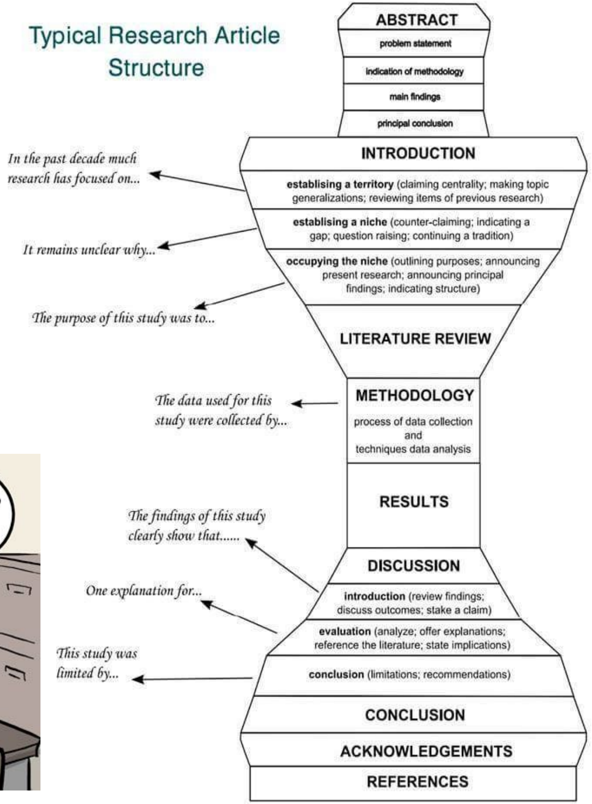


Image source : [LibGuides: Occupational Therapy: Research Article Structure]

Next Week

- Basics of searching for literature and giving a technical talks by Prof. Naumann