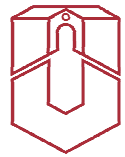
Natural Computation for Reasoning and Learning Applications

Kai-Uwe Kühnberger
(joint work with Helmar Gust, Peter Geibel, Angela Schwering, Ulf Krumnack)

University of Osnabrück,
Institute of Cognitive Science (IKW),
Artificial Intelligence Group
kkuehnbe@uos.de

School of Computer Science
University of St. Andrews
March 2009



Propaganda: The IKW

- The history of the Institute of Cognitive Science (IKW):
  - 1998: The international Bachelor's program Cognitive Science has been started (main teaching language is English).
  - 2002: The IKW and the international Graduate School *Cognitive Science* were founded. Both institutions are strongly oriented towards interdisciplinary research.
  - 2003: The current working groups have been established: Computational Linguistics, Artificial Intelligence, Cognitive Psychology, Philosophy of Mind, Neurobiopsychology, Neuroinformatics.
  - The institute is in the process of establishing new working groups: Cognitive Modeling, Biologically Oriented Computer Vision, Neuro/Psycholinguistics.
  - Since 2007 the IKW employs 3 assistant professors, 2 associate professors, and 3 full professors.

Some Features of the IKW

- Each year approx. 90 freshmen start their Cognitive Science education.
- Approx. 30 Master students join the IKW per year.
- Currently we have around 25 PhD students.
- Very recently we received a grant for 13 additional PhD positions.
- Since 2002 the IKW received in total significantly more than 10 Mio. Euros of research grants.
- Important research topics at the IKW:
  - Reinforcement learning / RoboCup
  - Attention research (biology oriented)
  - Text technology
  - Philosophy of emotion
  - Statistical NLP

Outline

- Challenges for Classical Approaches to Reasoning and Learning
- Learning and Reasoning with Analogies
  - Examples
  - Anti-Unification
  - Heuristic-Driven Theory Projection
- Learning Logic with Neural Networks
  - See Lecture Tomorrow
- Learning to Classify Documents
  - Motivation
  - Kernels for DOM Trees
  - Experiments
- Conclusions

Challenges for Classical Reasoning

- The manifold of reasoning paradigms used in applications led to the development of various corresponding formalisms.

Types of Reasoning	**Corresponding Formalisms**
Deductions	Classical Logic
Inductions	Inductive Logic Programming
Abductions	Extensions of Logic Programming
Analogical Reasoning	SME, LISA, AMBR, HDTP
Non-Monotonic Reasoning	Answer Set Programming
Frequency-Based Reasoning	Bayesian Reasoning
Vague and Uncertain Rasoning	Fuzzy, Probabilistic Logic
Etc.	Etc.

Challenges for Classical Reasoning

- Further difficulties for classical reasoning paradigms are summarized in the following list:
  - *Complexity problems:* most reasoning formalisms are either undecidable or exponential with respect to their complexity.
  - *Robustness:* Classical forms of reasoning are not very robust if only noisy or incorrect data is available.
  - *Profusion of representation formalisms:* Most reasoning types require specific representation formalisms that are often not simple to integrate.
  - *Integration:* A cognitive architecture that is intended to model intelligence on a human scale would need an integration and dynamic interplay between these different forms of reasoning.
- Non-classical forms of reasoning are needed for AI applications.

Challenges for Classical Learning

- The manifold of learning paradigms used in applications leads to the development of various corresponding formalisms. The following table summarizes some of the well-known learning paradigms.

	Learning Types	**Learning Approaches**
Unsupervised Learning	Clustering	Neural Networks, SOMs, ART, RBF network etc.
Supervised Learning	Classification, Learning a Function	Case-based reasoning, *k*-Nearest Neighbor, Decision Tree Learning
Reinforcement Learning	Policy Learning	*Q*-Learning, POMDPs, Temporal Difference Learning
Analytical & Inductive Learning	Rule Extraction, Learning in Domain Theories	Inductive Learning, Explanation-Based Learning, KBANNs

Further Challenges for Learning and Reasoning

- Approaches for reasoning and learning are facing further problems:
  - Learning from *noisy* data
  - Learning from *sparse* data
  - *Complexity* issues concerning learning
  - *Integration* of various learning mechanisms
- In order to overcome these problems, non-classical approaches for reasoning and learning are needed.
- In particular, *natural computation* has been proposed as an alternative.

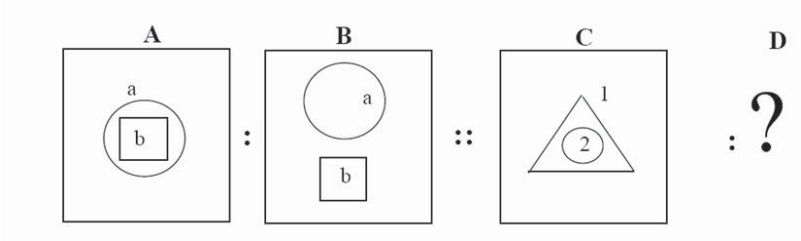
Natural Computation

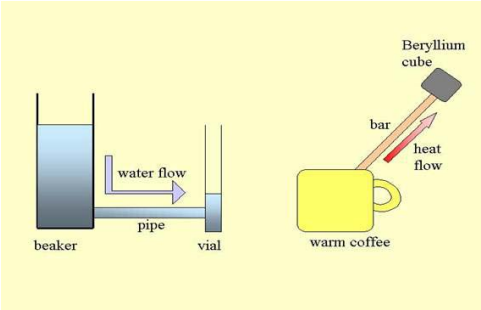
- I will take the term *natural computation* rather loosely.
- Here are examples of natural computation paradigms for reasoning and learning:
  - Learning with neural networks
  - Learning with support vector machines
  - Evolutionary computing
  - Dynamical systems
  - Etc.
- Besides these rather clear cases of *natural computation* there are some *cognitively inspired* approaches that could be used.
  - Analogical reasoning
  - Model-based reasoning and learning
  - Heuristic-based approaches to rationality
  - Etc.

Apology

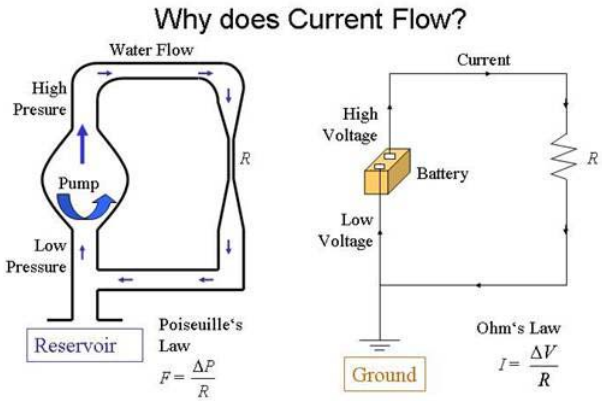
- Although it is clearly not possible to give solutions to the sketched problems, the following slides sketch some directions where resolutions of these problems could be found in the future.

Examples of Analogical Reasoning





⇔



Analogy as Abstraction

- The working hypothesis is that an analogical relation specifies a common structure in different domains.
- For example, in the Rutherford analogy *"Electrons are the planets of the nucleus"* we have the following corresponding common structures:
  - A massive objects attracts a lightweight object.
  - This causes the lightweight object to revolve around the heavier object.
  - Generalization: Central body (force) system.
- In the present framework, it is assumed that the source and target domains are given by (first-order) axioms.
- The idea of using anti-unification is based on the idea that the axiomatizations can be translated into each other via a generalization (anti-instance).

Anti-Unification

- Unification makes terms equal by the application of appropriate substitutions: applying the substitution $\Theta = \{y \leftarrow x, z \leftarrow x\}$ to the terms $p(f(x), y, g(y))$ and $p(f(x), z, g(x))$ results in

$$p(f(x), x, g(x))$$

- Anti-unification, introduced in Plotkin (1970) is the dual construction: input terms are t_1 and t_2, the output is a generalized term t such that for substitutions Θ_1 and Θ_2 it holds:

$$t_1 = t\Theta_1 \text{ and } t_2 = t\Theta_2.$$

Anti-Unification

- With respect to analogical reasoning the task is to find the least general generalizations of facts and rules for the target and source domains.
- It is well-known that for first-order anti-unification the following facts hold (Plotkin, 1970):
  - A generalization always exists.
  - There are at most finitely many generalizations.
  - There exists a unique least general generalization.
- What about second order anti-unification?
- If two terms $t_1 = f(a, b)$ and $t_2 = g(a, b)$ are given, a natural second-order generalization would be $F(a, b)$ (where F is a function variable) with substitutions $\Theta_1 = \{F \leftarrow f\}$ and $\Theta_2 = \{F \leftarrow g\}$. Then:

$$t_1 = f(a, b) = F(a, b)\Theta_1 \text{ and } t_2 = g(a, b) = F(a, b)\Theta_2$$

Heuristic-Driven Theory Projection

- HDTP is a formal approach for analogical reasoning based on many-sorted first-order logic.
- What is HDTP doing?
  - Input
    - First-order theories of the source and target domain.
  - Process
    - Select terms, predicates, formulas from target and source according to a heuristics.
    - Select best generalization according to a heuristics.
    - Transfer formulas from source to target if they are not associated yet.
  - Output
    - Generalized theory

Two Challenges of Analogy Making

- Challenge 1:
  - Second-order anti-unifications can result in several anti-instances. If not restricted at all, even infinitely many anti-instances are possible.
- Challenge 2:
  - It is rather implausible to assume that the domain theories are given in an appropriate form. Often one needs to re-represent the given axioms by logical deductions.

Resolving Challenge 1

- Restricted higher-order anti-unification is based on the following set of basic substitutions:
  - A *renaming* $\rho^{F,F'}$ replaces a variable $F \in \mathcal{V}_n$ by another variable $F' \in \mathcal{V}_n$ of the same argument structure:

$$F(t_1, \ldots, t_n) \xrightarrow{\rho^{F,F'}} F'(t_1, \ldots, t_n).$$

  - A *fixation* ϕ_c^V replaces a variable $F \in \mathcal{V}_n$ by a function symbol $f \in \mathcal{C}_n$ of the same argument structure:

$$F(t_1, \ldots, t_n) \xrightarrow{\phi_f^F} f(t_1, \ldots, t_n).$$

  - An *argument insertion* $\iota_{V,i}^{F,F'}$ with $0 \leq i \leq n$, $F \in \mathcal{V}_n$, $G \in \mathcal{V}_k$ with $k \leq n - i$, and $F' \in \mathcal{V}_{n-k+1}$ is defined by

$$F(t_1, \ldots, t_n) \xrightarrow{\iota_{V,i}^{F,F'}} F'(t_1, \ldots, t_{i-1}, G(t_i, \ldots, t_{i+k-1}), t_{i+k}, \ldots, t_n).$$

  - A *permutation* $\pi_\alpha^{F,F'}$ with $F, F' \in \mathcal{V}_n$ and bijective $\alpha : \{1, \ldots, n\} \to \{1, \ldots, n\}$ rearranges the arguments of a term:

$$F(t_1, \ldots, t_n) \xrightarrow{\pi_\alpha^{F,F'}} F'(t_{\alpha(1)}, \ldots, t_{\alpha(n)}).$$

Remarks

- If anti-unification is restricted to the four basic substitutions, one can show the following facts:
  - All first-order anti-unifications can be computed by combining the four basic substitutions.
  - In the second-order case, there are at most only finitely many anti-instances.
- Applying the theory of anti-unification to analogy problems requires some further steps:
  - Not only terms need to be anti-unified but also atomic formulas.
  - Whole theories need to be anti-unified.
  - For choosing appropriate pairs of candidate formulas for anti-unification heuristics are necessary.

Resolving Challenge 2

- Re-representation requires the consideration of the axioms Ax of a given set of input axioms, but also inferences contained in the theory $Th(Ax)$.
  - **Definition:** Let G be a finite set of formulas.
    - We call G an *anti-instance* of a set of formulas F iff there exists a substitution σ such that $Th(apply(G, \sigma)) \subseteq Th(F)$.
    - Let Ax_S, Ax_T be sets of formulas and σ, τ substitutions. We call the triple $\langle G, \sigma, \tau \rangle$ a *generalization* of Ax_S and Ax_T iff $Ax_S \xleftarrow{\sigma} G \xrightarrow{\tau} Ax_T$.
  - **Definition:** Given a generalization $\langle G, \sigma, \tau \rangle$ of Ax_S and Ax_T, the subset $Th(apply(G, \sigma))$ of $Th(Ax_S)$ is said to be *covered* by G and for Ax_T accordingly.
  - **Fact:** An anti-unifier $\langle G, \sigma, \tau \rangle$ has at least the same coverage as $\langle G', \sigma', \tau' \rangle$ if there exists a substitution $G' \xrightarrow{\theta} G$ that is compatible with the domain substitutions (i.e. $\sigma' = \sigma \circ \theta$ and $\tau' = \tau \circ \theta$).
- As a rule of thumb: “the greater the coverage the better”.

Integrating Various Forms of Reasoning

- An integration of various types of learning and reasoning can be achieved by HDTP:
  - The computed generalization is a form of learning that yields a structural description of the domains on an abstract level.
  - The transfer of knowledge from source to target can introduce new concepts in the target domain, i.e. learning takes place.
  - An iteration of analogical associations leads to an inductive refinement.
  - A re-representation tool allows the computation of deductive inferences.

References

- HDTP is described in [Gust, Kühnberger & Schmid, 2006, *TCS*].
- The syntactic principles of HDTP are examined in depth in [Schwering, Krumnack, Gust & Kühnberger, in press, *Cognitive Systems Research*].
- Restricted higher-order anti-unification was first introduced in [Krumnack, Schwering, Gust & Kühnberger, 2007, *AI'07*].
- Re-representation in HDTP was first introduced in [Krumnack, Gust, Kühnberger & Schwering, 2008, *AAAI'08*].
- Model constructions for the HDTP approach can be found in [Gust, Krumnack, Kühnberger & Schwering, 2007, *EuroCogSci'07*].
- Aspects of effective learning by HDTP can be found in [Gust & Kühnberger, 2006, *CogSci'06*].

Neural-Symbolic Integration

- A learning method that is in a strong sense "natural computation" is learning logic with neural means, more precisely, learning a model of a logical theory by neural networks.
  - Important work on this tradition was delivered by Steffen Hölldobler, Pascal Hitzler, Artur d'Avila Garcez, Dov Gabbay, Ekaterina Komendantskaya and many others.
- Neural-symbolic integration is the attempt to bridge the gap between subsymbolic and symbolic processing:
  - The data structure "logical interpretation function" as well as deduction steps in a proof are highly complex.
  - The potential input for a neural network is therefore heterogeneous.
  - Learning a closure of a logical axiomatic theory means learning the semantics of this theory.

Neural-Symbolic Integration

See Lecture Tomorrow

General Remarks

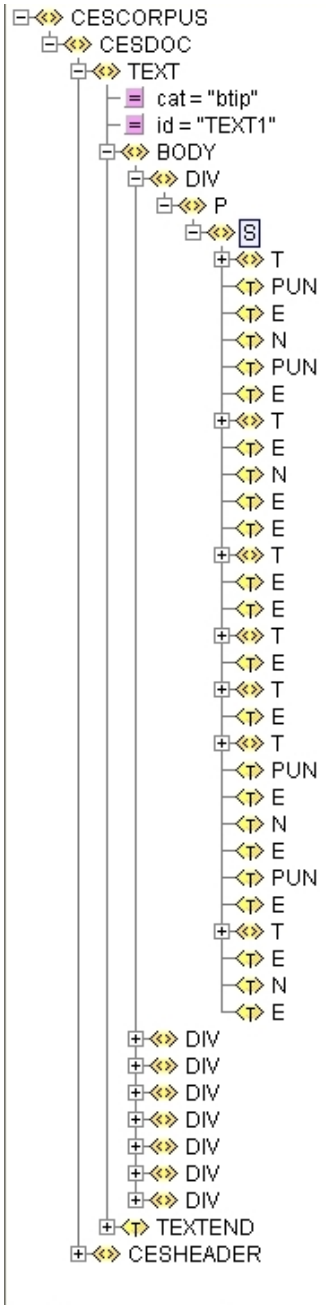
- This section discusses strategies to learn text genres on a purely structural basis, i.e. no coding of semantic knowledge is used.
  - The work of this section is based on [Geibel et al., 2008] and [Geibel, Gust & Kühnberger, 2007].
  - The learning strategy is based on kernel methods.
- Content, structure, and form of texts vary with communicative situations or the respective function of texts.
- Example: text genres [Martin, 1992; Ventola, 1987].
- The working hypothesis is: structural differences reflect functional ones, while similar functions tend to result in similarly structured texts.
- To which degree is class membership manifested by *structural* properties?

Machine Learning Perspective

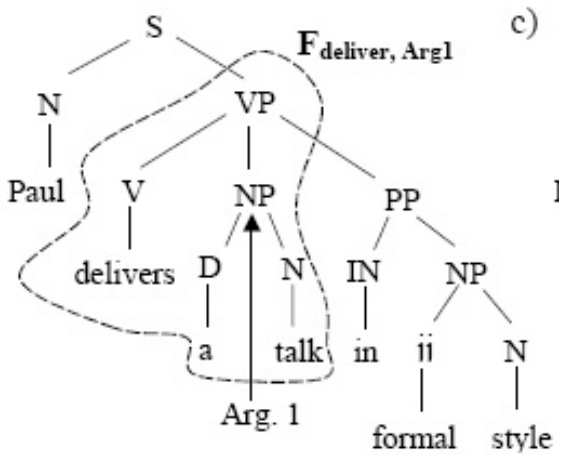
- Idea: Is it possible to learn to classify text types on purely structural properties?
- We restrict our considerations to *trees*.
- Examples are DOM trees of XML documents, i.e. we consider logical document structures.
- Support Vector Machines have proved to be useful for text classification in the past [Joachims, 1998].
- *Tree Kernels* [Collins & Duffy, 2002] can be used for classifying DOM trees:
  - Parse tree kernels are useful for trees generated by a grammar [cf. Moschitti, 2004].
  - $\rightarrow$ *How can we make this applicable for DOM trees?*
  - Kaschima & Konayagi (2002) and Moschitti (2006) both presented tree kernels based on a combination with string kernels.
  - $\rightarrow$ *Are they applicable for large corpora?*

Trees for XML Documents

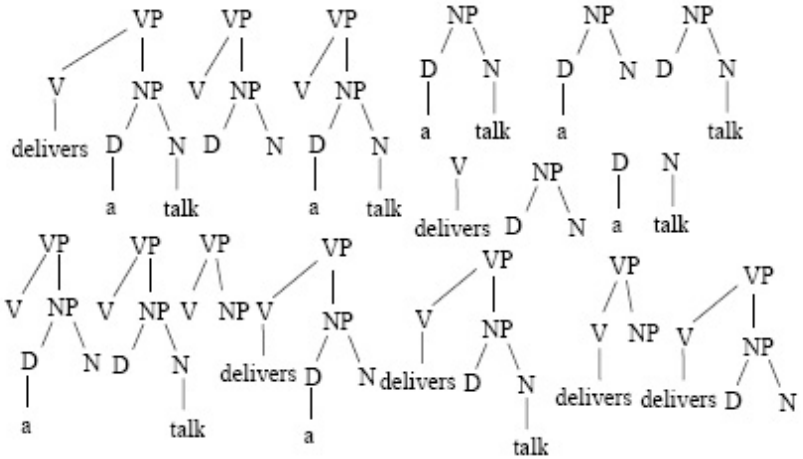
- XML files can be represented by their DOM trees (Document Object Model).
- DOM trees are rooted, labeled, ordered trees.
- Inner nodes are labeled with XML tags.
- Leaves might be labeled with sentences (maybe also represented as trees).
- Approaches work for every tree-like presentation.



The Parse Tree Kernel [Collins & Duffy]



Paul delivers a talk in formal style.



- Semantic role labelling [Moschitti, 2004]:
  - Given: a parsed tree
  - Classify subtree as: logical subject, logical object, locations, ...
- How can we achieve a feature-based representation of a tree?
- A tree is described by all possible intermediate parse trees t_i resulting in a feature vector $(\phi_{t_1}, \ldots, \phi_{t_k})$.

The Parse Tree Kernel [Collins & Duffy]

- Assume $\Delta(v, v')$ is defined as the number of isomorphic mappings of partial parse trees rooted in nodes v and v'.
- Collins & Duffy (2002) showed that the kernel can be computed recursively by the formula $k(T, T') = \sum_{v \in V, v' \in V'} \Delta(v, v')$ such that:
  - $\Delta(v, v') = 0$ if the productions applied in v and v' are different.
  - $\Delta(v, v') = 1$, if the productions in v and v' are identical and both nodes are pre-terminals.
  - For other non-terminals with identical productions:

$$\Delta(v, v') = \prod_{i=1}^{n(v)} (1 + \Delta(v_i, v_i'))$$

Variants of Kernels

- Using these ideas it is possible to define new variants of tree kernels that can be used for text classification.
- The simple tree kernel takes pairs of labels into account: incorporate a kernel k^{Σ} operating on pairs of node labels (tags, attributes, text).
- The left-aligned tree kernel is based on the idea: Compare just as many children as possible, if the number of children differs.
- The set tree kernel treats children as a set.
- The soft tree kernel use a soft comparison of node positions using a RBF kernel.

Artificial Tree Data

Three classes with 10 examples each:

1. The class 1 examples all have a left-aligned subtree of the form $g(a, b(e), c)$.
2. The class 2 examples all have a general ordered subtree of the form $g(c, b, e(a))$, where gaps are allowed but the ordering of the subtrees c, b and $e(a)$ has to be preserved.
3. The class 3 examples contain subtrees of the form $g(c, b, a(e))$, where the child trees c, b and $a(e)$ are allowed to occur reordered and gaps might have been inserted, too.

Artificial Tree Data

Class 1:
f(n,h(m),g(a,b(e),c))
f(h(m,g(a,b(e),c)))
f(g(a,b(e),c,n),h(m))
f(g(a,b(e),c,n,m))
f(a,m,m,h(h(g(a,b(e),c))),n)
f(g(a,b(e),c),g(e(a),c,a))
f(h(m,c),g(a,b(e),c),g(m,n,m))
f(g(a,b(e),c,h(h(m),n,b)))
f(h(h(g(a,b(e),c))))
f(g(a,b(e),c),a)

Class 2:
f(n,h(n),g(c,m,b,n,e(a)))
f(h(h(m),n,b),g(c,b,n,m,e(a)))
f(h(g(h(n),c,b,e(a))),b)
f(h(g(n,c,b,e(a))),h(m,b,h(a,n)))
f(g(b(e),c,a),g(c,n,b,e(a)))
f(g(c,h(c,n),b,h(h(h(b))),e(a)))
f(g(c,b,e(a)),g(m,b,n,a,b(e),n,c))
f(b,g(c,b,e(h(b,m,a),a)))
f(g(g(a),c,d,m),g(c,b,h(n),e(a)))
f(g(m,c,b,e(a),h(a)))

Class 3:
f(n,g(c,n,b,m,a(e)),h(m))
f(m,e,b,h(g(b,n,c,m,a(e))))
f(h(h(a(e),m,b,n,n,c)),b,e)
f(e,b,g(m,c,a,b,e,e,a(e)))
f(b,e,g(a(e),m,b,h(a),m,c))

Class 3 (continued):
f(g(h(m,e),a(e),h(b),c,a,b))
f(g(b,h(m),c,h(n),a(b,e,a)))
f(g(m,h(n,h(n)),g(b,e,c,a(e))))
f(g(a(e),g(h(n)),b,c),b,n)
f(h(h(e),h(n),b,h(n),m),g(c,m,n,b,a(e)

Results

Table: Optimal F-Measures

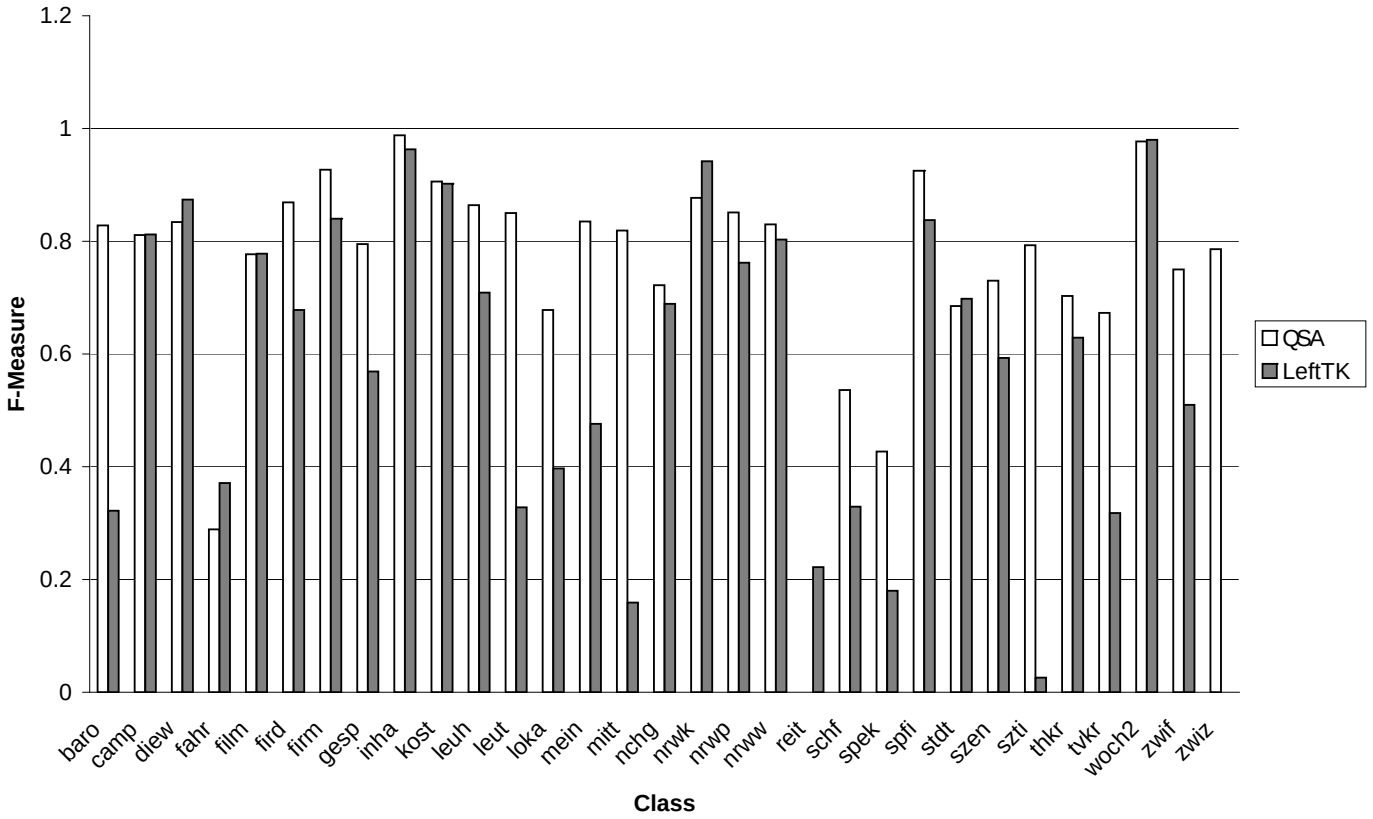
	Class 1	Class 2	Class 3
TagTK	0.727	0.6	0.736
LeftTK	0.909	0.363	0.44
SetTK	0.952	1.00	1.00
SoftTK	1.0	1.0	1.0
StringTK	1.0	1.0	1.0

- TagTK is based on XML tags only (like bag-of-words).
- The implementations of the SoftTK and the StringTK perform best.

Süddeutsche Zeitung Corpus

- Approx. 35000 short texts from Süddeutsche Zeitung were considered.
- 31 Classes:
  - Bühnentip (theater)
  - Hochschulnachrichten (university news)
  - Fragen und Antworten (questions and answers)
  - Dates
  - Inhalt (content)
  - Wochenchronik (chronicle of the week)
  - Etc.
- Document description
  - Document DOM tree + class.

Experiments



- QSA often better than LeftTK, but not in every case.
- LeftTK trained only on downsampled data set.
- Many classes are learnable, some still have problems.
- SoftTK and SetTK: bad results.

General Conclusions

- Some challenges for classical forms of reasoning and learning were sketched:
  - Variety of reasoning and learning mechanisms.
  - Complexity problems
  - Robustness
  - Profusion of representation formalisms
  - Lack of integration devices
- The claim is that certain non-classical types of reasoning and learning can help in addressing certain aspects of these challenges.
- Structure learning with kernel methods can help to classify text types.
  - Structural information can be used to learn something about the content of a text (to a certain extent).

General Conclusions

- Analogical reasoning can be tool for the integration of various reasoning methods.
- Furthermore, analogies can be used to explain learning from sparse data and certain aspects of creativity.
- Neural-symbolic integration can be used to learn logical theories and bridge the gap between symbolic and subsymbolic approaches (more on that tomorrow).
  - Integration of different computing paradigms
  - Integration of reasoning and learning
  - Robustness

Thank you very much !

The Simple Tree Kernel SimTK

- The parse tree kernel is only applicable if there is a grammar available.
- Trees consisting of only a single leave are excluded.
- It is not possible to include node labels.
- $\rightarrow$ Incorporate kernel k^{Σ} operating on pairs of node labels (tags, attributes, text).
- If there are either no children, or the number of children differs we set

$$\Delta_{\text{SimTK}}(v, v') = \lambda \cdot k^{\Sigma}(\alpha(v), \alpha(v'))$$

Else:

$$\Delta_{\text{SimTK}}(v, v') = \lambda \cdot k^{\Sigma}(\alpha(v), \alpha(v'))(1 + \prod_{i=1}^{n(v)} \Delta_{\text{SimTK}}(v_i, v_i'))$$

- λ is a weight for tree depth.

Pros and Cons of SimTK

- PROS of the SimTK:
  - No grammar is presupposed.
  - We can include complex node labels, e.g. text in leave nodes by plugging in an appropriate “sub”-kernel.
- Shortcomings of the SimTK:
  - If the number of children differs, then the children are not compared at all.
  - If the number of children is not different, then they are only compared in the original order.
- In texts of the same type, however, children might be permuted, modified, missing, or inserted.

The Left-Aligned Tree Kernel

- Compare just as many children as possible, if the number of children differs.
- Recursive case:

$$\Delta(v, v') = \lambda \cdot k^{\Sigma}(\alpha(v), \alpha(v'))\left(1 + \sum_{k=1}^{\min(n(v), n'(v'))} \prod_{i=1}^{k} \Delta(v_i, v_i')\right)$$

- Concerning the feature space we allow general subtrees of a tree T, with the restriction that subtrees are left-aligned.
- Pros:
  - LeftTK is more flexible than SimTK regarding length.
- Cons:
  - Trees occurring more on the left have a higher influence than trees occurring on the right.
  - Still no permutations are allowed.

The Set Tree Kernel

- The idea to circumvent the last point: treat children as a set.
- Recursion

$$\Delta(v, v') = \lambda \cdot k^{\Sigma}(\alpha(v), \alpha(v'))\left(1 + \sum_{i=1}^{n(v)} \sum_{i'=1}^{n'(v')} \Delta(v_i, v'_{i'})\right)$$

- A suitable feature space interpretation are rooted label sequences (for $k^{\Sigma} = k^{id}$ and $\lambda = 1$).
- Pros:
  - SetTK allows a variable number of children.
  - SetTK is more flexible than LeftTK regarding ordering.
- Cons:
  - No information about ordering retained at all.
- Idea: keep some ordering information.

The Soft Tree Kernel

- Idea: Use a fuzzy / soft comparison of node positions using RBF kernel:

$$k_\gamma(x, y) = e^{-\gamma(x-y)^2}$$

- Recursion:

$$\Delta(v, v') = \lambda \cdot k^{\Sigma}(\alpha(v), \alpha(v')) \cdot k_\gamma(\mu(v), \mu'(v')) \cdot \left(1 + \sum_{i=1}^{n(v)} \sum_{i'=1}^{n'(v')} \Delta(v_i, v'_{i'})\right)$$

- Pros:
  - Has everything that is necessary.
- Cons:
  - There is a new parameter to tune: γ

The String Tree Kernel (StringTK)

- Alternative: treat child tree sequences as string.
- The idea is similar to Kashima & Konayagi (2002) and Moschitti (2006), respectively.
- Minor improvements:
  - Incorporates a gap penalty, in contrast to Kashima & Konayagi (2002), but similar to Moschitti (2006).
  - To control complexity a parameter L for the maximum substring length is introduced.
- Pros:
  - Has everything and is elegant
- Cons:
  - Has the worst complexity: $O(|V| \cdot |V'| \cdot b \cdot b \cdot L)$
- Which is best for what?

Quantitative Structure Analysis (QSA)

- We represent each text as a vector of structure features.
- Structure Level: S is the set of constituent types, e.g., sentence, paragraph, phrase.
- Features: Each $s_i \in S$ is described with respect to a set of features F_j.
- F_j may represent, for example, the *complexity* (i.e., the number of immediate daughter elements) or the *length* (i.e., the number of leafs dominated by) of a corresponding instance of s_i.
- Description of a text: This is done by means of parameters of location or statistical spread.
- Pros: efficient to compute.
- Cons: no complex structural properties.