Dependency Tree Kernels For Relation Extraction From Natural Language Text

Frank Reichartz, Hannes Korte & Gerhard Paass

Fraunhofer IAIS, Sankt Augustin, Germany

Introduction

- Relation Extraction is the task of identifying **semantic relations** between entities within the same sentence
- Most approaches consider binary target relations

Examples

"Recently Obama became the president of the USA."

Role(Obama, USA)

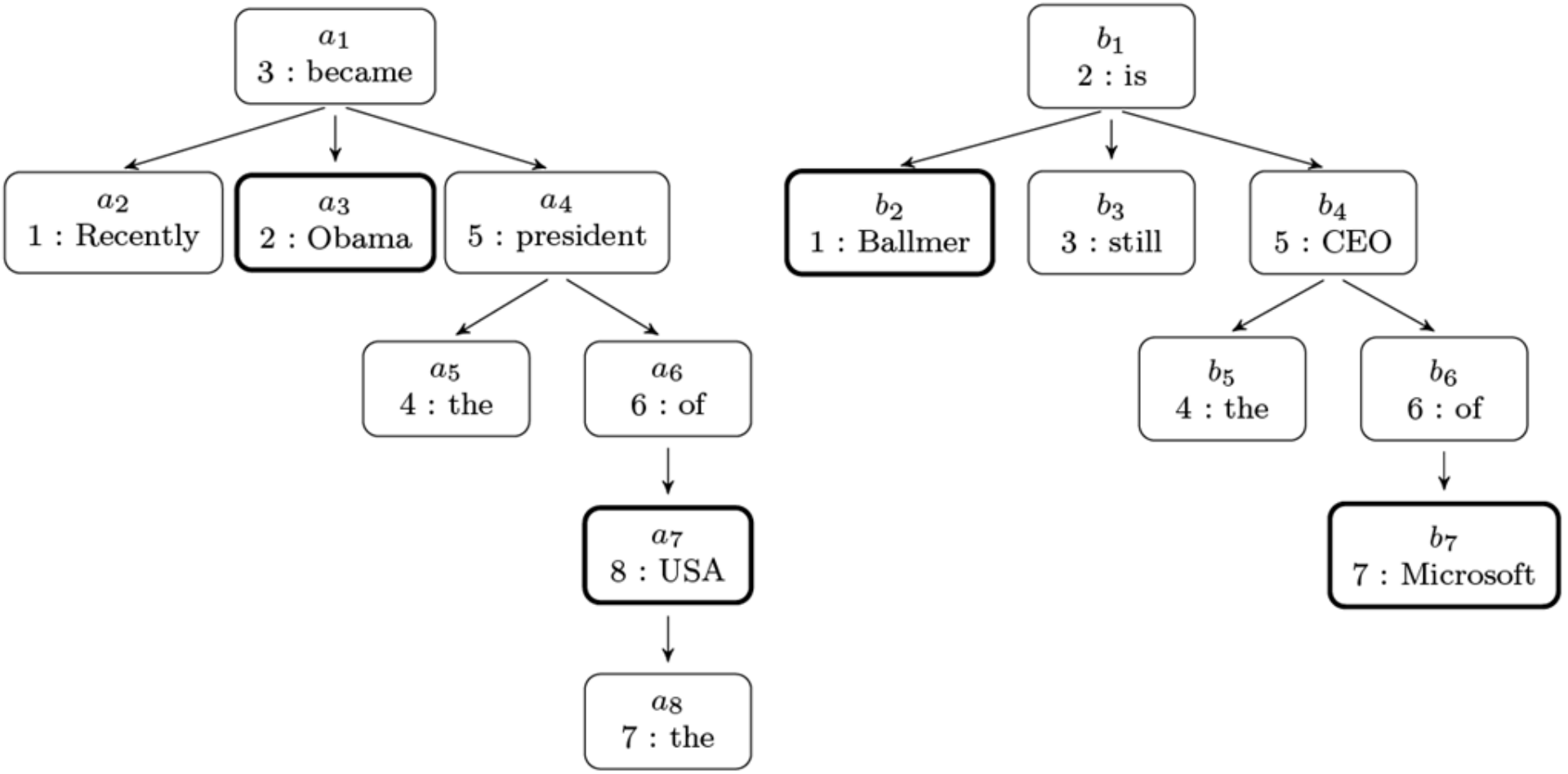
"Ballmer is still the CEO of Microsoft."

Role(Ballmer, Microsoft)

Parse Trees

- Syntactial Parsers transform natural language sentences into **tree structures**
  - Extensive information on **syntactic structure**
  - Variety of tree types in NLP e.g. dependency, phrase grammar etc.
- Focus here: **Dependency Parse Trees**
  - A dependency tree is a structured representation of the grammatical dependency between the words of a sentence by a **vertex labeled directed tree**
  - **Bijection** between words and vertices (implies ordering of children)
  - Good for language with free word order e.g. German
  - Can be generated with a high quality by freely available parsers

Dependency Trees

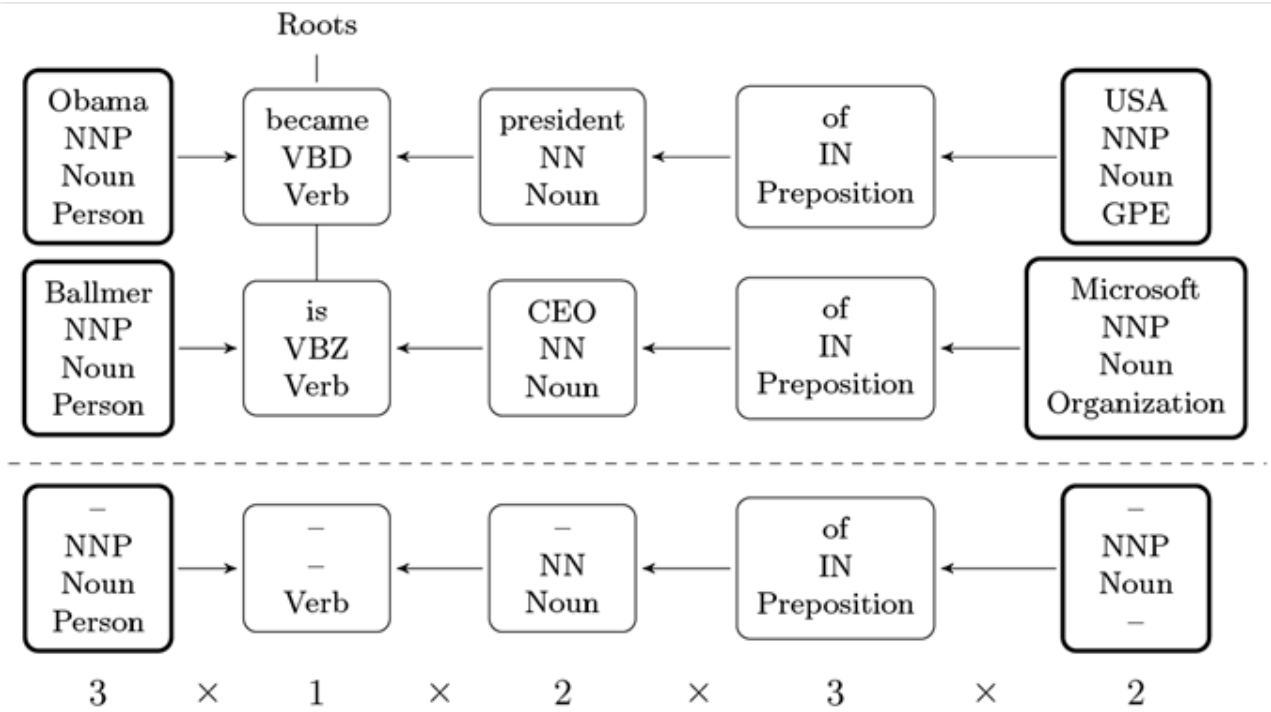


Kernels

- Generalized dot product for linear classifiers which implicitly maps the observations into a higher-dimensional feature space
  - A dot product in feature space can be expressed by a kernel working on the input space (kernel trick)
  - This input space can consist of structured data like parse trees. In this setting the embedding into a high-dimensional space may only be implicit.

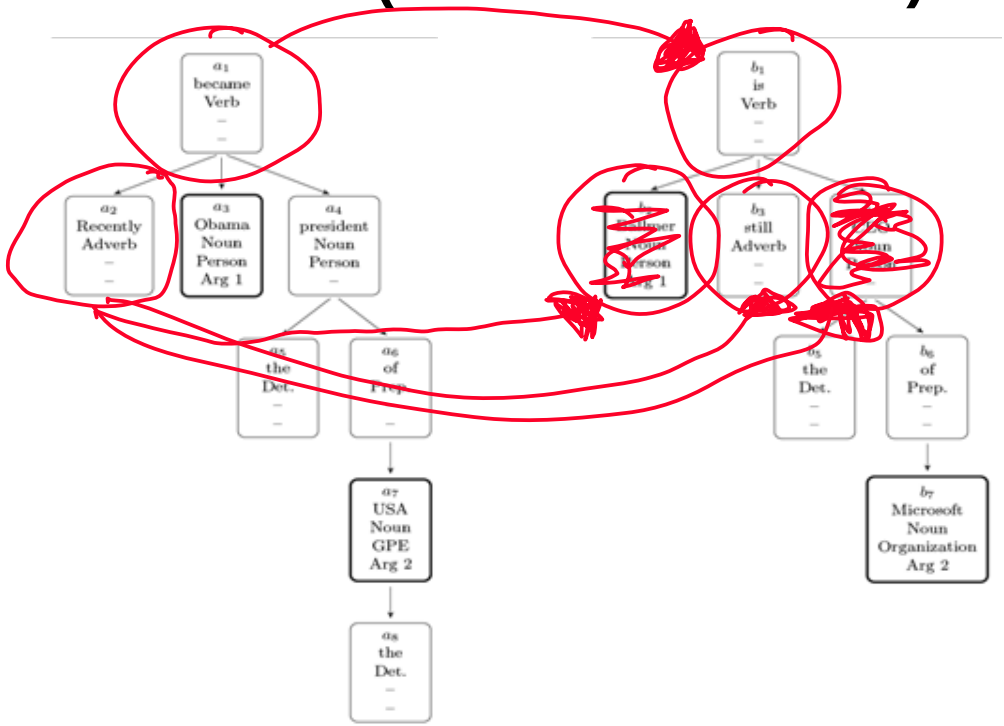
- Usage of kernels over parse trees as structured information for RE has been shown to be useful:
  - Culotta et al. 2004 , Bunescu & Mooney 2005 (Dependency Parse Trees)
  - Zang et al. 2008 (Phrase Grammar Parse Trees)

Shortest Path Kernel (Bunescu & Mooney 2005)



- Efficient Computation
- Node similarity function
- „same“ length restriction ~ very restrictive

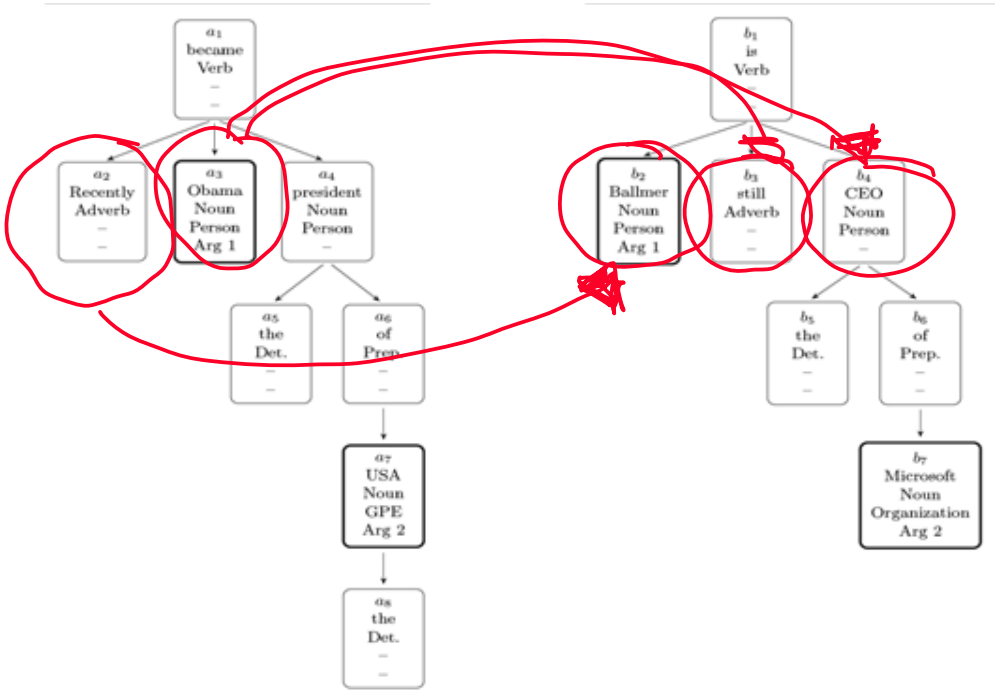
Dependency Tree Kernel (Culotta et al. 2004)



- Node similarity & node matching function
- One „no match“ = „zero value“

Fraunhofer IAIS

Dependency Tree Kernel (Culotta et al. 2004)



- Node similarity & node matching function
- One „no match“ = „zero value“

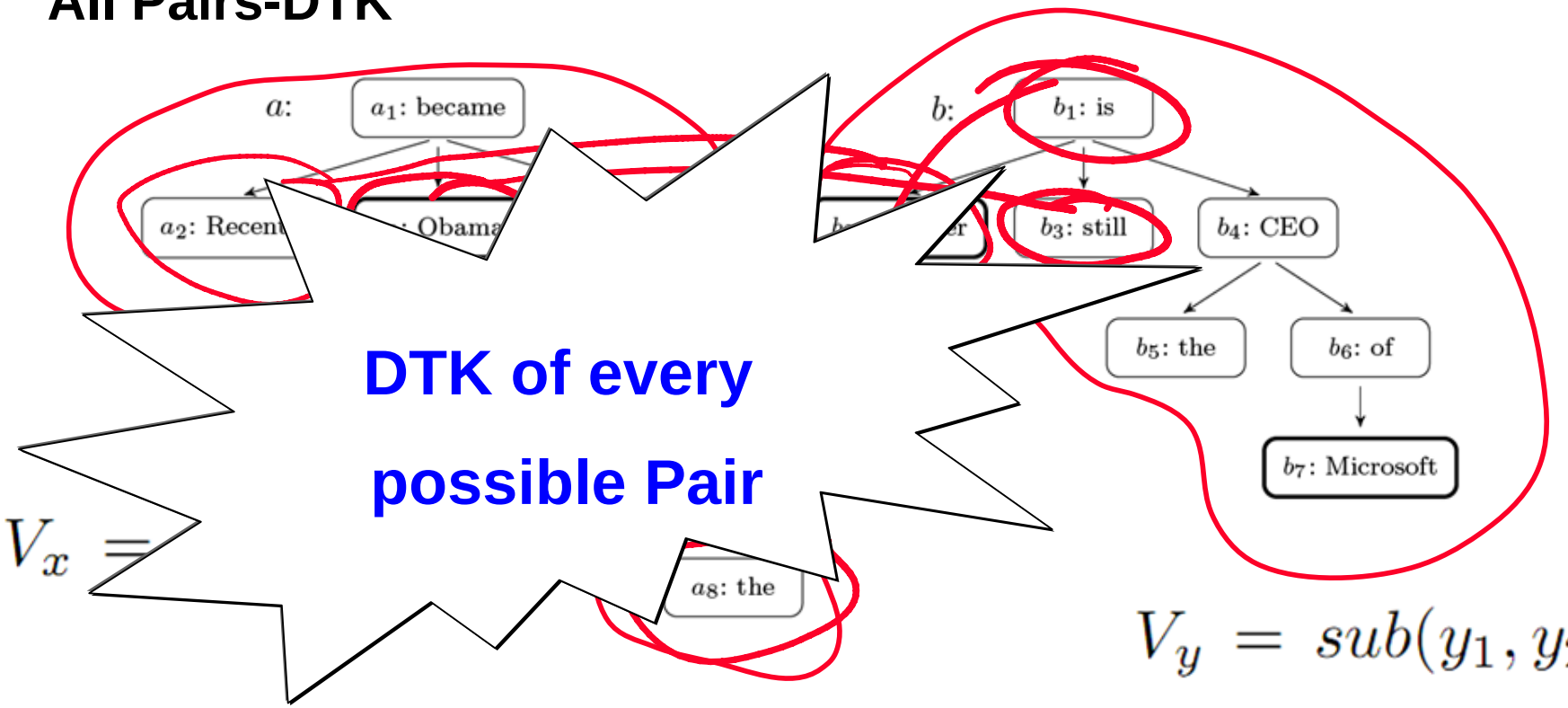
Moitivation Of Our Approaches

- DTK only considers the root nodes (of the subtrees) and their matching children
  - Possibly discarding similar structures at lower levels of the trees

- First Approach:

  Apply the DTK to **every combination of nodes** in the subtrees implied by the relation nodes

All Pairs-DTK



$V_x =$

$V_y = sub(y_1, y_2)$

$$K_{\text{All-Pairs}}(X, Y) = \sum_{u \in V_x} \sum_{v \in V_y} \Delta(u, v)$$

DTK

Fraunhofer IAIS

Motivation Of Second Approach

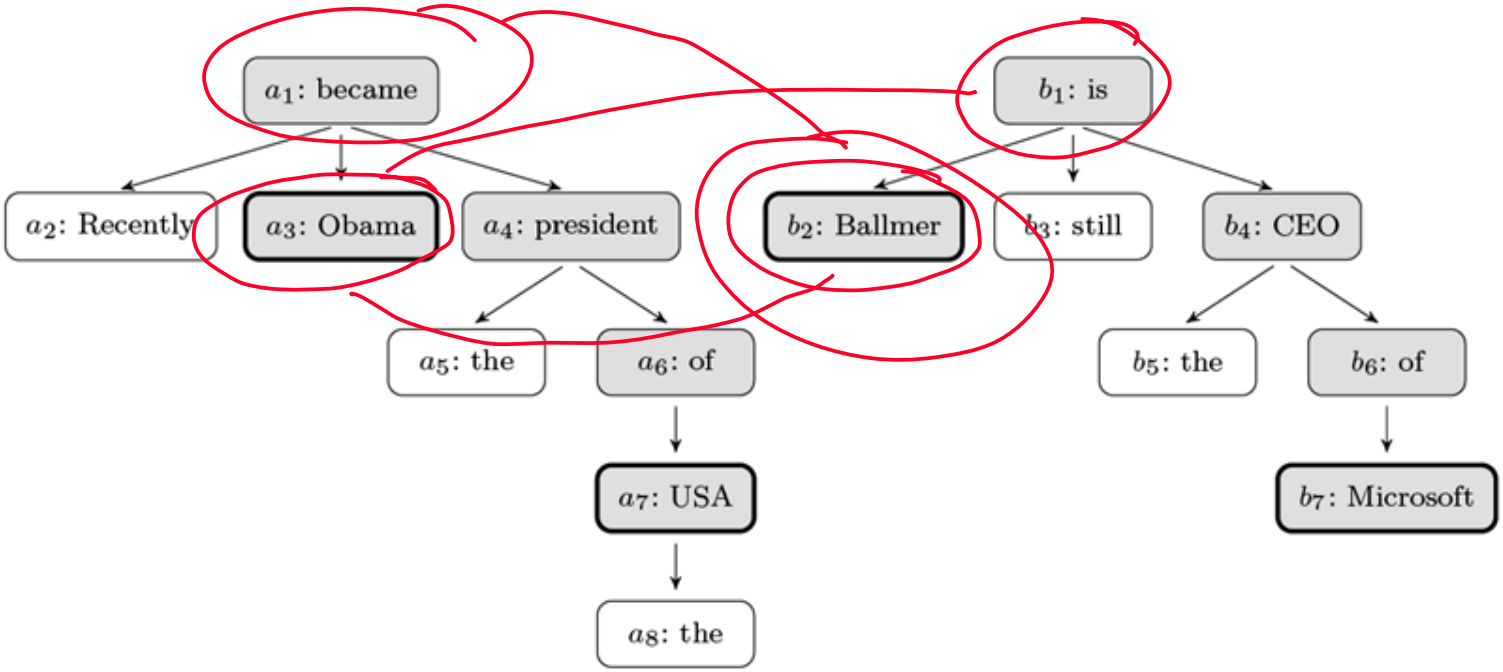
- **Shortest Path Hypothesis** (Bunsecu & Mooney)
  - Most valuable information on the path between two entities
  - Also valuable Information close to this Path
- Second Approach:

  Consider the information below and on the path in parallel

➢ Apply the DTK to the path node sequences instead of only to both root nodes

  - Relaxing the same length restriction of the SPK by using **all possible subsequences** of nodes on the path

Path-DTK



- Considers all subtrees on the path computing a similarity for each pair
- **Efficient computation** by intelligent caching exploiting DT statistics

$$K_{\text{Path-DTK}}(X,Y) = \sum_{i \in I_{|x|},\, j \in I_{|y|},\, |\mathbf{i}|=|\mathbf{j}|,\, d(\mathbf{i}),d(\mathbf{j}) \leq q} \mu^{d(\mathbf{i})+d(\mathbf{j})} \quad \Delta'(x(\mathbf{i}), y(\mathbf{j})) \quad M(x(\mathbf{i}), y(\mathbf{j}))$$

subsequences of same lenth upto lengt q

Penalty for gaps

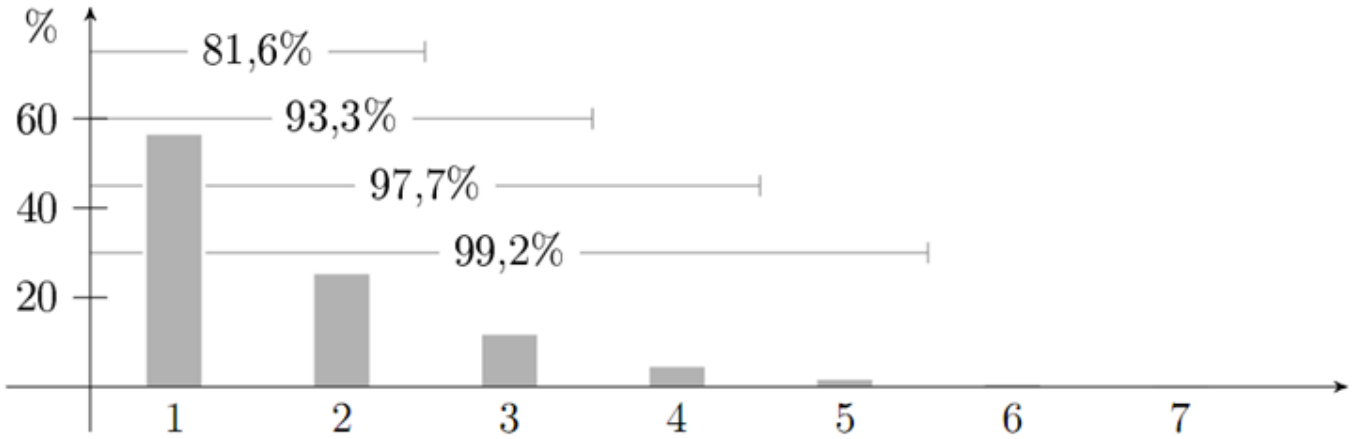
IAIS

Efficient Computation of the DTK

- Culotta et al. applied the ideas of the String (subsequence) Kernel of Lodhi et al. for the computation of the child subsequences in $\mathrm{O}(mn^3)$
- We adapted a later $\mathrm{O}(mn^2)$ **optimization** (Christinini and J. Shawe-Taylor) to this problem
- But: A closer look into the dataset reveals that the Dependency tree nodes have very few children

Efficient Computation of the DTK

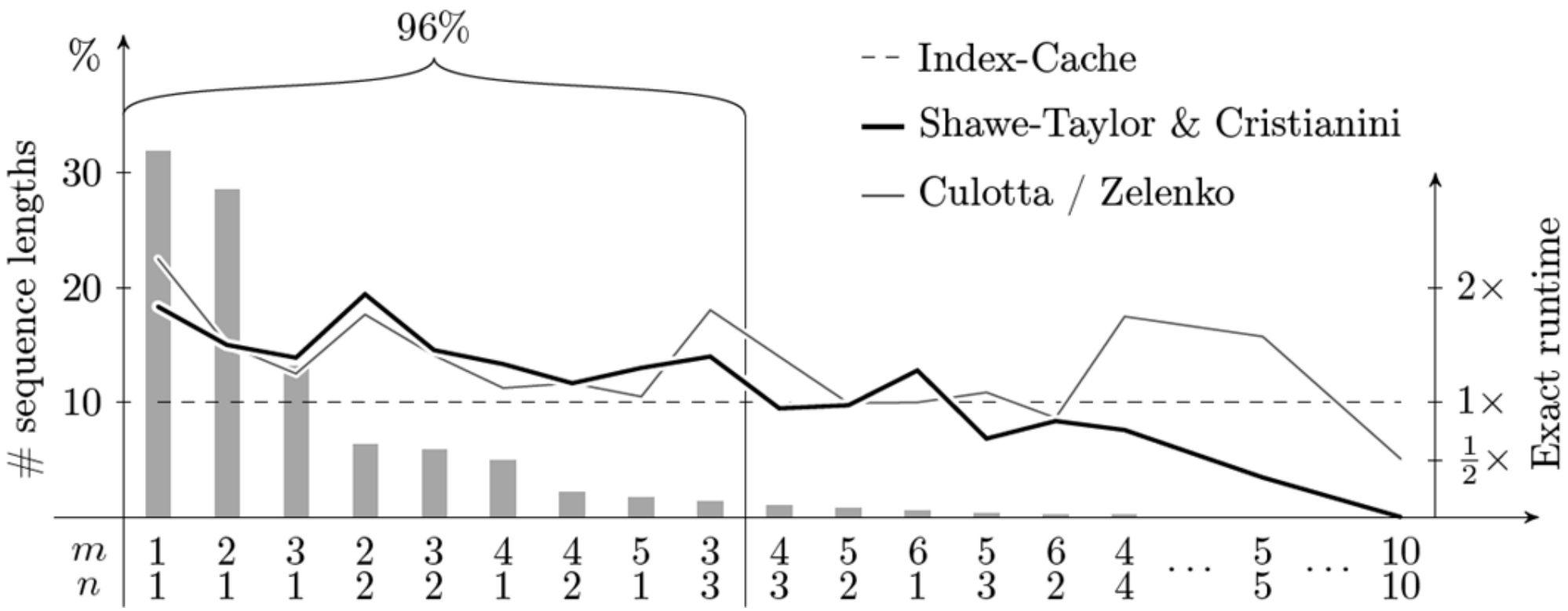
- The out-degrees of the nodes have the following Distribution:



- The upper bounds hold for large values of m and n
- We propose a different approach by **efficient caching**

Efficient Computation of the DTK

- Child sequence length combinations (m>n)



Fraunhofer IAIS

Empirical Evaluation

- Experiments conducted on ACE-2003 dataset
  - Public benchmark dataset for Relation Extraction
    - –Newspaper texts: entities and relations ~10k trees
  - 5 Top Level Relation
    - –Role, Social, Near, At and Part
  - Experiments
    - –5-times repeated 5-fold cv
    - –Pre-specified Split

Relation	Named–Named # Training	# Test
At	481	106
Near	44	14
Part	265	64
Role	732	155
Social	55	4

Empirical Results

Cross-validation

Kernel	q	μ	λ	C	Precision	Recall	F_{micro}	F_{macro}
SPK [2]	–	–	–	1	79.8%	46.4%	58.6%(0.14)	34.2%(0.14)
DTK [4]	–	–	0.65	100	71.7%	53.7%	61.4%(0.32)	44.5%(0.63)
All-Pairs-DTK	–	–	0.6	60	73.1%	57.8%	64.5%(0.26)	**57.7%**(0.54)
Path-DTK	1	0.5	0.5	10	**80.2%**	**61.2%**	**69.4%**(0.09)	57.3%(0.40)

Pre-specified Training/Test Split

Kernel	q	μ	λ	C	Precision	Recall	F_{micro}	F_{macro}
SPK [2]	–	–	–	1	74.7%	34.4%	47.1%	25.0%
DTK [4]	–	–	0.65	100	79.5%	44.0%	56.7%	36.9%
All-Pairs-DTK	–	–	0.6	60	**80.2%**	49.6%	61.3%	40.6%
Path-DTK	1	0.5	0.5	10	76.7%	**52.8%**	**62.5%**	**41.3%**

Results on Cross-Validation

Kernel	Relation	Precision	Recall	F-score	Std. error
SPK [4]	At	78.1%	41.8%	54.5%	(0.34)
	Near	10.0%	0.5%	0.9%	(0.78)
	Part	76.8%	25.4%	38.1%	(0.76)
	Role	81.4%	62.9%	71.0%	(0.16)
	Social	54.0%	3.6%	6.8%	(0.06)
DTK [2]	At	65.2%	47.4%	54.9%	(0.52)
	Near	52.9%	29.1%	37.3%	(2.20)
	Part	71.8%	41.8%	52.8%	(0.67)
	Role	78.4%	67.1%	72.3%	(0.30)
	Social	11.5%	3.6%	5.5%	(0.73)
All-Pairs-DTK	At	65.2%	54.0%	59.1%	(0.28)
	Near	93.2%	71.8%	**80.9%**	(1.57)
	Part	70.0%	43.4%	53.6%	(0.48)
	Role	79.1%	67.8%	73.0%	(0.44)
	Social	39.6%	15.6%	**22.2%**	(2.08)
Path-DTK	At	73.4%	58.0%	**64.8%**	
	Near	90.9%	50.5%	64.9%	
	Part	85.5%	49.8%	**62.9%**	(0.67)
	Role	83.4%	71.9%	**77.2%**	(0.13)
	Social	42.4%	10.6%	16.8%	(1.34)

Path-DTK better for Relations with more data

Conclusion / Future Work

- Performance depends on **type of relation** and **training size**
- For specific relations sufficient performance for **real applications**
- Combination with phrase grammar parse tree kernels reasonable (Reichartz et al. ACL ‘09)

- Open Source version of our SVM-Kernel toolkit for structured data
- More sophisticated node-similarity functions
- Combination of ML-Approaches (e.g. pattern recognition)
- Application to other languages (e.g. German)

The End

Thank You!

Questions?