The Impact of Training Data Division in Inductive Dependency Parsing

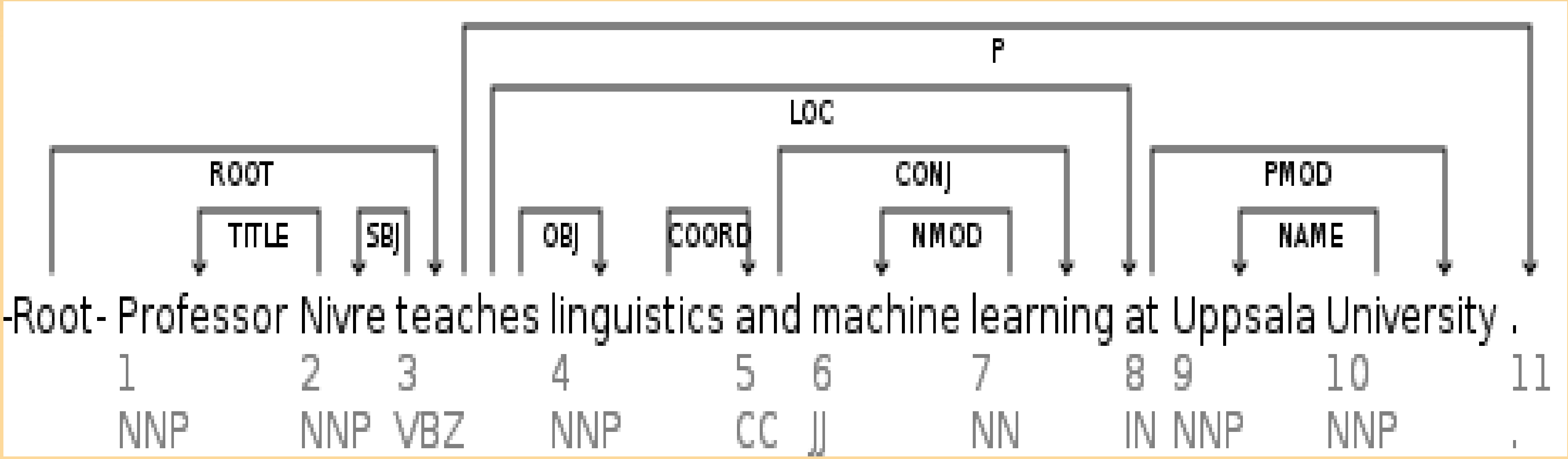
- Kjell Winblad
- Internal supervisor Olle Gällmo
- External supervisor at:
- The Department of Linguistics and Philology
- Joakim Nivre

Dependency Parsing

- The process of parsing sentences to create a graph with words as vertices and edges with labels as representations of dependencies between the words.
- Grammars using this form to describe the structure of sentences are called dependency grammars

Dependency Grammar

- The dependency structure can be used in for example automatic translation systems
- Sentence with dependency structure:



Inductive Dependency Parsing

- To automatically generate a ”good” dependency structure for a sentence is a hard problem because of the complexity of natural languages and the ambiguity of words.
- Promising results have been archived by using machine learning methods

Inductive Dependency Parsing

Transition-based parsing system

- A configuration
- A set of rules
- An oracle function (constructed with machine learning method or from a grammar)

The algorithms apply the rule given by the oracle function given the current configuration, until the sentence is parsed

Inductive Dependency Parsing

Training

- Extract the necessary rules to construct the correct dependency tree from example sentences.
- The result is a set of elements of the form *configuration=>rule* that can be represented as an vecor of numbers
- The converted training data can be used in any standard machine learning system to train an oracle function

Machine Learning Methods

- Most accurate result: Nonlinear Support Vector Machines (SVM)

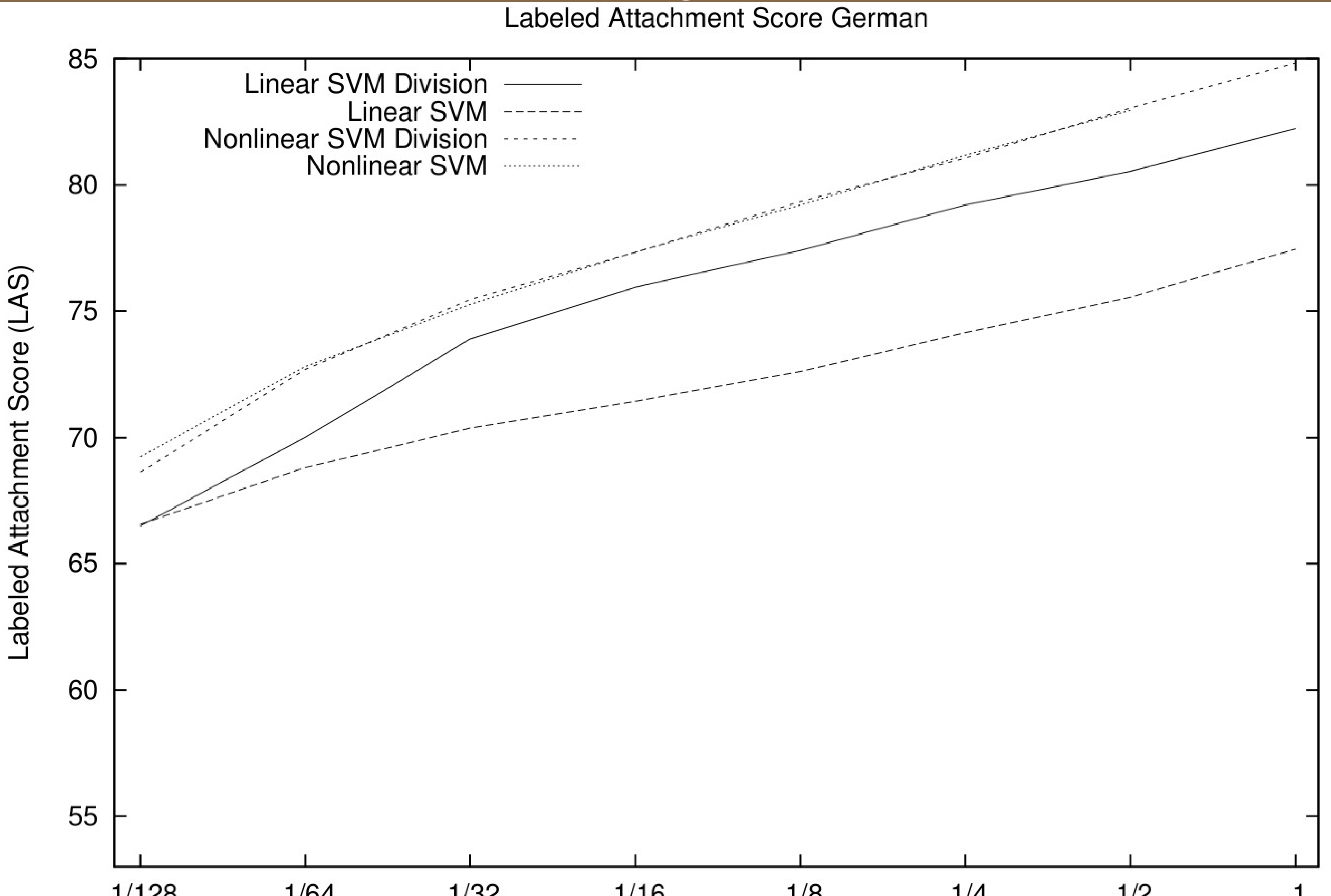
  - Slow training

  - Require a huge amount of memory
- Worse: Linear SVMs

  - Fast training and parsing
- Better: Divide the training data by a particular feature before training with a Linear SVM

  - Still fast training and parsing

Machine Learning Methods Test



Machine Learning Methods Test

Linear SVM

Size		1/128	1/64	1/32	1/16	1/8	1/4	1/2	1
...									
German Div	TR	0.02	0.03	0.07	0.16	0.35	0.69	1.36	2.51
	TE	0.02	0.02	0.03	0.03	0.05	0.06	0.10	0.15
	AC	66.50	70.02	73.89	75.94	77.40	79.21	80.54	82.24
German	TR	0.02	0.03	0.07	0.21	0.39	0.90	1.83	3.26
	TE	0.03	0.03	0.02	0.03	0.03	0.03	0.03	0.04
	AC	66.56	68.82	70.38	71.44	72.61	74.15	75.54	77.45

Nonlinear SVM

Size		1/128	1/64	1/32	1/16	1/8	1/4	1/2	1
...									
German Div	TR	0.05	0.08	0.11	0.21	1.18	3.70	17.81	77.91
	TE	1.24	1.37	0.75	0.80	1.65	2.32	4.13	7.59
	AC	68.64	72.70	75.45	77.33	79.35	81.07	83.05	84.82
German	TR	0.07	0.24	1.34	5.31	23.03	98.02	420.84	–
	TE	2.72	3.83	6.52	13.10	20.72	32.82	58.99	–
	AC	69.25	72.81	75.26	77.34	79.21	81.19	82.96	–

Objectives

- Find out why dividing the training set gives better accuracy when a linear SVM is used
- Is it possible to improve the accuracy even further while still keeping the fast training and parsing time with a more advanced division strategy?

Goal

- Find a division strategy that can divide the training data in an optimal way:
- Maximize:

accuracy_improvement - generalization_error

Experiment: Divide worst partions with another feature

	Language	Size	Feat. 1	Feat. 2	No div.
Worse Than Average	Swedish	0.52	86.90	86.25	86.82
	Chinese	0.64	89.50	89.39	89.57
	German	0.39	88.10	87.91	85.84
Better Than Average	Swedish	0.48	95.72	95.59	95.72
	Chinese	0.36	96.75	96.43	96.61
	German	0.61	95.54	95.41	94.93
Everything	Swedish	1.0	91.15	90.75	90.92
	Chinese	1.0	92.09	91.72	92.04
	German	1.0	92.68	92.52	91.04

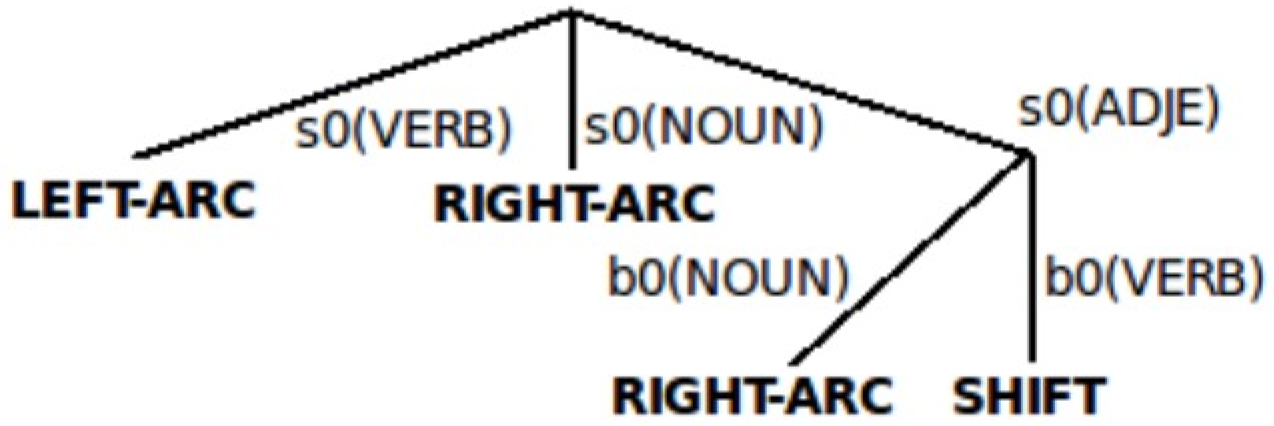
Exepriment: Dividing even more

	No Div.	Sign.	1 Div.	Sign.	2 Div.
Swedish	90.916	$<$ (70%)	91.150*	$>$ (55%)	90.979
Chinese	92.044	$<$ (21%)	92.089	$<$ (36%)	92.168*
German	91.039	$<$ (99%)	92.678	$<$ (22%)	92.708*
Czech Stack Lazy	89.979	$<$ (99%)	91.175	$<$ (99%)	91.852*
Czech Stack Projection	89.783	$<$ (99%)	91.016	$<$ (99%)	91.616*
English Stack Lazy	94.374	$<$ (99%)	94.756	$<$ (99%)	94.954*
English Stack Projection	94.400	$<$ (99%)	94.763	$<$ (99%)	95.008*
Average	*91.791*		*92.518*		*92.755**

Experiment: Decision Tree

Top of stack	First in buffer	Rule
VERB	ADJE	**LEFT-ARC**
NOUN	ADJE	**RIGHT-ARC**
ADJE	NOUN	**RIGHT-ARC**
ADJE	VERB	**SHIFT**

Table 2.1: Training examples for the decision tree example.

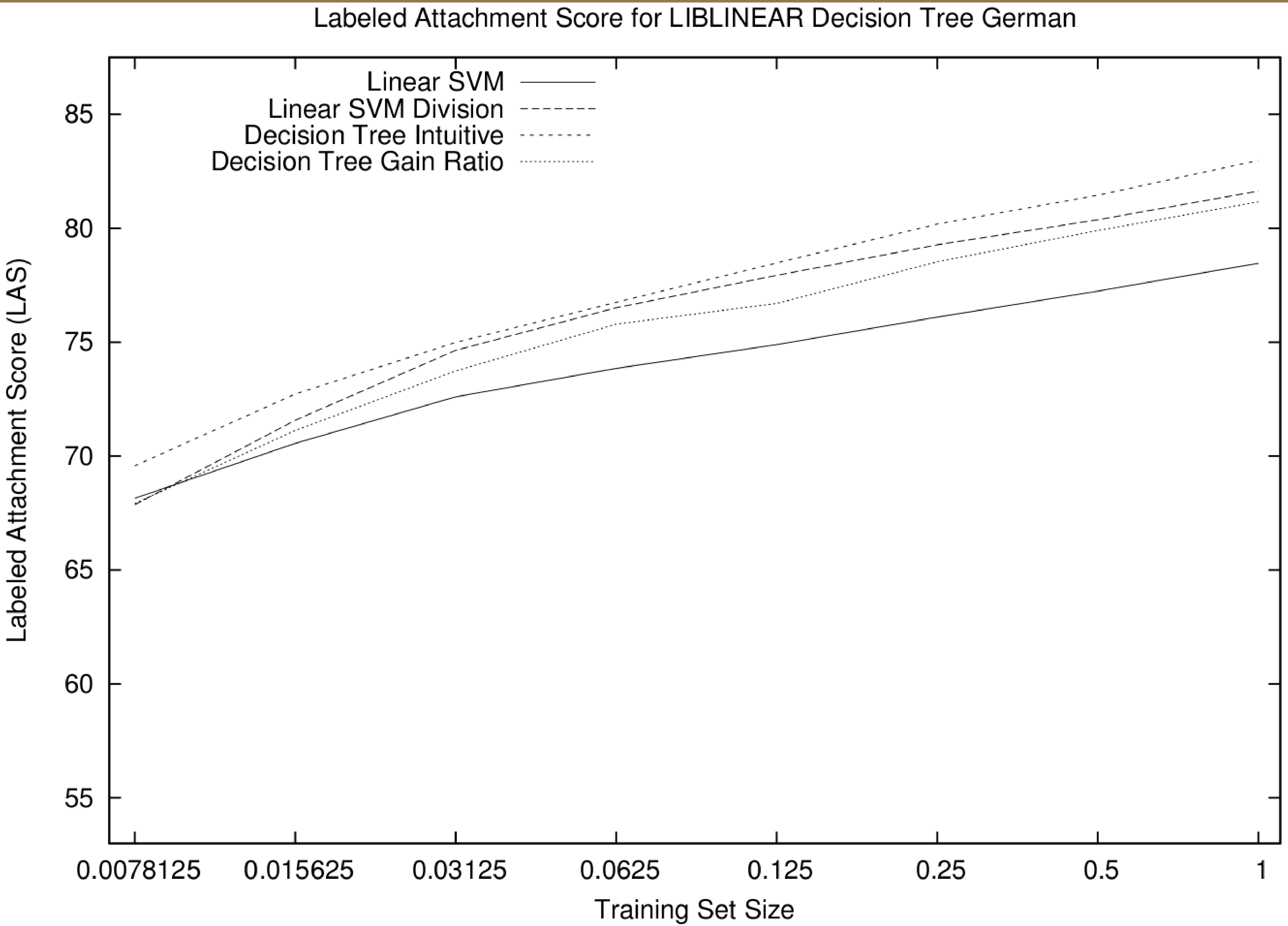


Decision Tree Creation

- Divide the training data
- If the division improve the accuracy then perform the algorithm for all new nodes
- The test is done with cross validation

Decision Tree Results

	Intuitive	Sign.	2 Div.	Sign.	Gain Ratio
Swedish	91.168	> (60%)	90.979*	> (11%)	90.947
Chinese	92.118	< (23%)	92.168*	> (15%)	92.135
German	93.132*	> (99%)	92.708	< (96%)	92.936
Czech Stack Lazy	91.866	> (10%)	91.852	< (63%)	91.947*
Czech Stack Projection	91.654	> (27%)	91.616	< (85%)	91.771*
English Stack Lazy	95.020	> (65%)	94.954	< (98%)	95.120*
English Stack Projection	95.077	> (67%)	95.008	< (96%)	95.158*
Average	*92.862**		*92.755*		*92.859*



Labeled Attachment Score for LIBLINEAR Decision Tree German
Linear SVM
Linear SVM Division
Decision Tree Intuitive
Decision Tree Gain Ratio
Labeled Attachment Score (LAS)
85
80
75
70
65
60
55
0.0078125
0.015625
0.03125
0.0625
0.125
0.25
0.5
1
Training Set Size

Decision Tree in MaltParser

Liblinear Decision Tree in MaltParser

Size		1/128	1/64	1/32	1/16	1/8	1/4	1/2	1
...									
German Division	TR	0.02	0.03	0.09	0.17	0.31	0.41	0.73	1.17
	TE	0.02	0.02	0.03	0.05	0.04	0.06	0.09	0.14
	AC	69.53	72.68	75.03	76.63	77.94	79.25	80.37	81.61
German Decision Tree Intuitive	TR	0.08	0.09	0.22	0.54	1.22	2.16	4.86	12.33
	TE	0.03	0.02	0.03	0.06	0.07	0.13	0.31	1.18
	AC	69.57	72.72	74.99	76.75	78.48	80.18	81.45	82.98
German Decision Tree Gain Ratio	TR	0.04	0.07	0.14	0.33	0.73	1.82	4.45	12.67
	TE	0.02	0.02	0.02	0.03	0.06	0.11	0.28	0.97
	AC	67.92	71.13	73.74	75.79	76.70	78.53	79.90	81.16

Conclusions

- Linear SVM + Division
  - Better accuracy than only a Linear SVM
  - Parallelizing easy with division
    - Better parsing and training time
- Nonlinear SVMs + Division
  - faster training and parsing
  - not much difference in accuracy

Conclusions

- Decision tree with linear SVMs in leafs

  - Improved accuracy slower training and parsing

  - Could potentially be more important when more training data is available

Future work

- Test the decision tree division strategy with more optimal feature extraction models

  - Largest obstacle is the memory requirements

  - Could be solved by parallelization

- Other variants of division trees would be interesting to test