Computerlinguistik II

Vorlesung im SoSe 2019
(M-GSW-10)

Prof. Dr. Udo Hahn

Lehrstuhl für Computerlinguistik
Institut für Germanistische Sprachwissenschaft
Friedrich-Schiller-Universität Jena

http://www.julielab.de

Two Paradigms for NLP

- Symbolic Specification Paradigm
  - Manual acquisition procedures
  - Lab-internal activities
  - Intuition and (few!) subjectively generated examples drive progress based on individual (competence) judgments
    - "I have a system that parses all of my nine-teen sentences!"

Symbolic Specification Paradigm

- Manual rule specification
  - Source: linguist´s intuition
- Manual lexicon specification
  - Source: linguist´s intuition
- Each lab has its own (home-grown) set of NLP software
  - Hampers reusability
  - Limits scientific progress
  - Waste of human and monetary resources (we “burnt” thousands of Ph.D. student all over the world ☹)

Shortcomings of the "Classical" Linguistic Approach

- Huge amounts of background knowledge req.
  - Lexicons (approx. 100,000 – 150,000 entries)
  - Grammars (>> 15,000 – 20,000 rules)
  - Semantics (>> 15,000 – 20,000 rules)
- As the linguistic and conceptual coverage of classical linguistic systems increases (slowly), it still remains insufficient; systems also reveal 'spurious' ambiguity, and, hence, tend to become overly "brittle" and unmaintainable
- More fail-soft behavior is required at the expense of ... ? (e.g., full-depth understanding)

Two Paradigms for NLP

- Symbolic Specification Paradigm
  - Manual acquisition procedures
  - Lab-internal activities
  - Intuition and (few!) subjectively generated examples drive progress based on individual (competence) judgments
    - “I have a system that parses all of my nine-teen sentences!”
- **Empirical (Learning) Paradigm**
  - **Automatic acquisition procedures**
  - **Community-wide sharing of common knowledge and resources**
  - **Large and ‚representative‘ data sets drive progress according to experimental standards**
    - **“The system was tested on 1,7 million words taken from the WSJ segment of the MUC-7 data set and produced 4.9% parsing errors, thus yielding a statistically significant 1.6% improvement over the best result by parser X on the same data set & a 40.3% improvement over the baseline system!”**

Empirical Paradigm

- Large repositories of language data
  - Corpora (plain or annotated, i.e., enriched by meta-data)
- Large, community-wide shared repositories of language processing modules
  - Tokenizers, POS taggers, chunkers, NE recognizers, ...
- Shared repositories of machine learning algos
- Automatic acquisition of linguistic knowledge
  - Applying ML algos to train linguistic processors by using large corpora with valid linguistic metadata (linguist as educated data supplier, „language expert“) rather than manual intuition (linguist as creative rule inventor)
- Shallow analysis rather than deep understanding
- Large, community-wide self-managed, task-oriented competitions, comparative evaluation rounds
- Change of mathematics:
  - Statistics rather than algebra and logics

Paradigm Shift – We Exchanged our Textbooks…

POS Tagging

A	severe	infection	ended	the	pregnancy	.
DET	ADJ	NOUN	VERB	DET	NOUN	ST

Penn Treebank Tag Set

Tag	Description	Examples
.	sentence terminator	. ! ?
DT	determiner	all an many such that the them these this
JJ	adjective, numeral	first oiled separable battery-powered
NN	common noun	cabbage thermostat investment
PRP	personal pronoun	herself him it me one oneself theirs they
IN	preposition	among out within behind into next
VB	verb (base form)	ask assess assign begin break bring
VBD	verb (past tense)	asked assessed assigned began broke
WP	WH-pronoun	that what which who whom

In total, 45 tags

Transformation Rules for Tagging [Brill, 1995]

- Initial State: Based on a number of features, guess the most likely POS tag for a given word:
  - die/DET Frau/NOUN ,/COMMA die/DET singt/VFIN
- Learn transformation rules to reduce errors:
  - *Change* DET *to* PREL *whenever the preceding word is tagged as* COMMA
- Apply learned transformation rules:
  - die/DET Frau/NOUN,/COMMA die/PREL singt/VFIN

First 20 Transformation Rules

	Change Tag		
#	From	To	Condition
1	NN	VB	Previous tag is *TO*
2	VBP	VB	One of the previous three tags is *MD*
3	NN	VB	One of the previous two tags is *MD*
4	VB	NN	One of the previous two tags is *DT*
5	VBD	VBN	One of the previous three tags is *VBZ*
6	VBN	VBD	Previous tag is *PRP*
7	VBN	VBD	Previous tag is *NNP*
8	VBD	VBN	Previous tag is *VBD*
9	VBP	VB	Previous tag is *TO*
10	POS	VBZ	Previous tag is *PRP*
11	VB	VBP	Previous tag is *NNS*
12	VBD	VBN	One of previous three tags is *VBP*
13	IN	WDT	One of next two tags is *VB*
14	VBD	VBN	One of previous two tags is *VB*
15	VB	VBP	Previous tag is *PRP*
16	IN	WDT	Next tag is *VBZ*
17	IN	DT	Next tag is *NN*
18	JJ	NNP	Next tag is *NNP*
19	IN	WDT	Next tag is *VBD*
20	JJR	RBR	Next tag is *JJ*

Taken from: Brill (1995), Transformation-Based Error-Driven Learning

Towards Statistical Models of Natural Language Processing …

Letter-based Language Models

- Shannon’s Game
- Guess the next letter:
-

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
- W

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
- Wh

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
- Wha

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
- What

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
- What d

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
- What do

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
- What do you think the next letter is?

Word-based Language Models

- Shannon’s Game
- Guess the next letter:
- What do you think the next letter is?
- Guess the next word:
-

Word-based Language Models

- Shannon's Game
- Guess the next letter:
- What do you think the next letter is?
- Guess the next word:
- We

Word-based Language Models

- Shannon's Game
- Guess the next letter:
- What do you think the next letter is?
- Guess the next word:
- We are

Word-based Language Models

- Shannon's Game
- Guess the next letter:
- What do you think the next letter is?
- Guess the next word:
- We are now

Word-based Language Models

- Shannon's Game
- Guess the next letter:
- What do you think the next letter is?
- Guess the next word:
- We are now entering

Word-based Language Models

- Shannon's Game
- Guess the next letter:
- What do you think the next letter is?
- Guess the next word:
- We are now entering statistical

Word-based Language Models

- Shannon's Game
- Guess the next letter:
- What do you think the next letter is?
- Guess the next word:
- We are now entering statistical territory

Approximating Natural Language Words

- zero-order approximation: letter sequences are independent of each other and all equally probable:
  - xfoml rxkhrjffjuj zlpwcwkcy ffjeyvkcqsghyd

Approximating Natural Language Words

- first-order approximation: letters are independent, but occur with the frequencies of English text:
  - ocro hli rgwr nmielwis eu ll nbnesebya th eei alhenhtppa oobttva nah

Approximating Natural Language Words

- second-order approximation: the probability that a letter appears depends on the previous letter
  - on ie antsoutinys are t inctore st bes deamy achin d ilonasive tucoowe at teasonare fuzo tizin andy tobe seace ctisbe

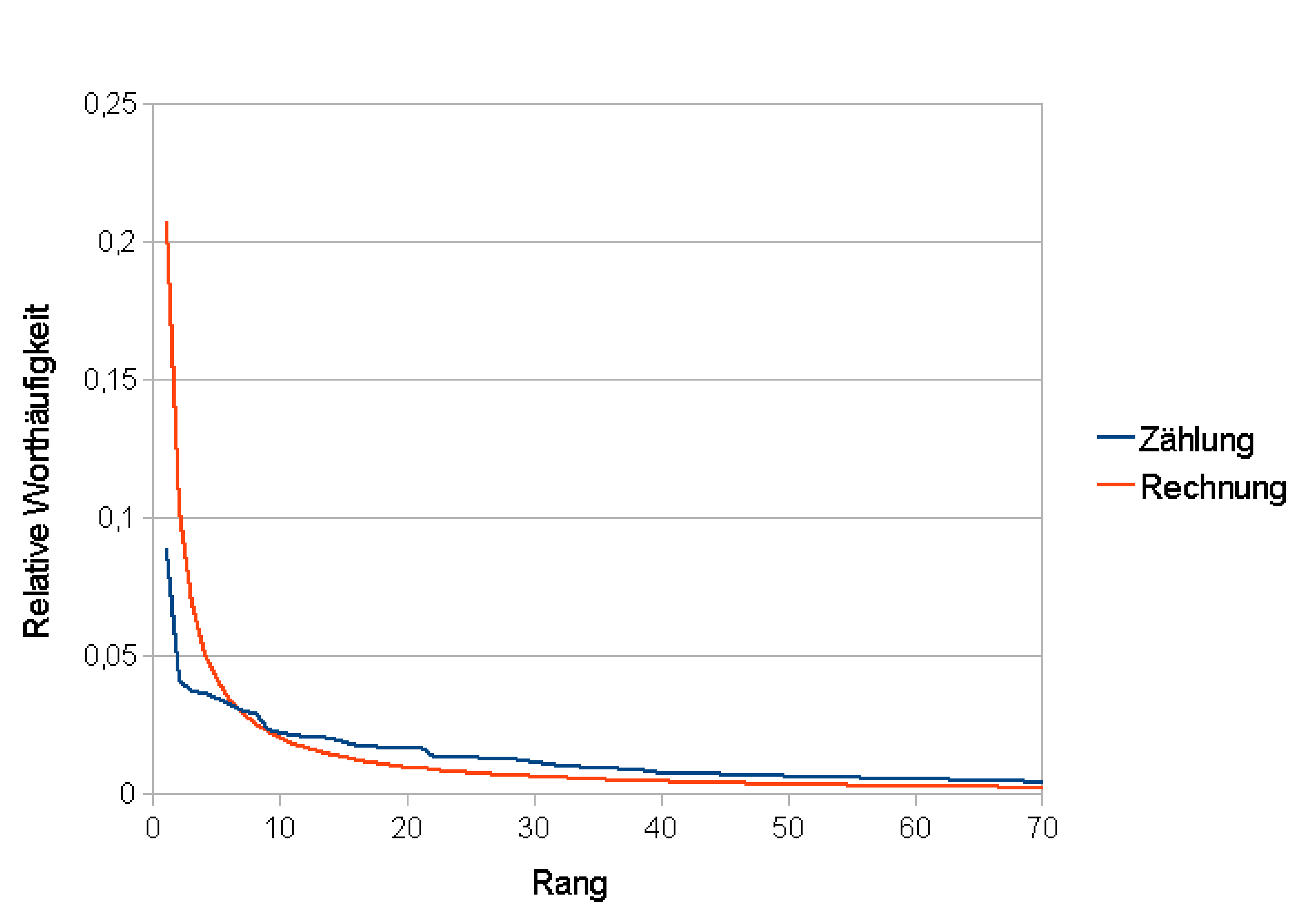
Approximating Natural Language Words

- third-order approximation: the probability that a certain letter appears depends on the two previous letters
  - in no ist lat whey cratict froure birs grocid pondenome of demonstures of the reptagin is regoactiona of cre

Approximating Natural Language Words

- Higher frequency trigrams for different languages:
  - English: THE, ING, ENT, ION
  - German: EIN, ICH, DEN, DER
  - French: ENT, QUE, LES, ION
  - Italian: CHE, ERE, ZIO, DEL
  - Spanish: QUE, EST, ARA, ADO

Zipfsches Gesetz



Wortverteilung im Vergleich zu einer einfachen Zipf-Verteilung (~1/n. Wortanzahl: 70; Texte aus: `http://www.gutenberg.org/dirs/etext04/8effi10.txt`)

Terminology

- **Sentence**: unit of written language
- **Utterance**: unit of spoken language
- **Word Form**: the inflected form that appears literally in the corpus
- **Lemma**: lexical forms having the same stem, part of speech, and word sense
- **Types (V)**: number of distinct words that might appear in a corpus (vocabulary size)
- **Tokens (N_T)**: total number of words in a corpus (note: $V << N_T$)
- **Types seen so far (T)**: number of distinct words seen so far in corpus (note: $T <_{=} V << N_T$)

Word-based Language Models

- A model that enables one to compute the probability, or likelihood, of a sentence S, P(S).
- Simple: Every word follows every other word with equal probability (0-gram)
  - Assume |V| is the size of the vocabulary V
  - Likelihood of sentence S of length n is $1/|V| \times 1/|V| \ldots \times 1/|V|$
  - If English has 100,000 words, the probability of each next word is 1/100000 = .00001

Relative Frequency vs. Conditional Probability

- Smarter: *Relative* Frequency

  Probability of each next word is related to word frequency within a corpus (unigram)

  - Likelihood of sentence S = $P(w_1) \times P(w_2) \times \ldots \times P(w_n)$
  - Assumes probability of each word is independent of probabilities of other words

Relative Frequency vs. Conditional Probability

- Smarter: *Relative* Frequency

  Probability of each next word is related to word frequency within a corpus (unigram)

  - Likelihood of sentence S = $P(w_1) \times P(w_2) \times \ldots \times P(w_n)$
  - Assumes probability of each word is independent of probabilities of other words

- Even smarter: *Conditional* Probability

  Look at probability given previous words (n-gram)

  - Likelihood of sentence S = $P(w_1) \times P(w_2|w_1) \times \ldots \times P(w_n|w_{n-1})$
  - Assumes probability of each word is dependent on probabilities of previous words

Generalization of Conditional Probability via Chain Rule

- Conditional Probability for Two Events, A_1 and A_2
  - $P(A_1,A_2) = P(A_1) \cdot P(A_2|A_1)$
- Chain Rule generalizes to multiple (*n*) events
  - $P(A_1, \ldots, A_n) =$

    $P(A_1) \times P(A_2|A_1) \times P(A_3|A_1,A_2) \times \ldots \times P(A_n|A_1 \ldots A_{n-1})$

    - Examples:
      - P(the dog) = P(the) × P(dog | the)
      - P(the dog bites) = P(the) × P(dog | the) × P(bites| the dog)

Relative Frequencies and Conditional Probabilities

- Relative word frequencies are better than equal probabilities for all words
  - In a corpus with 10K word types, each word would have P(w) = 1/10K
  - Does not match our intuitions that different words are more likely to occur
    - (e.g. “the” vs. “shop” vs. “aardvark”)
- Conditional probability is more useful than individual relative word frequencies
    - dog may be relatively rare in a corpus
    - but if we see barking, P(dog|barking) may be large

Probability for a Word String

- **In general, the probability of a complete string of words $w_1^n = w_1 \ldots w_n$ is**

$$P(w_1^n)$$

$$= P(w_1)P(w_2|w_1)P(w_3|w_1\ w_2)\ldots P(w_n|w_1\ldots w_{n-1})$$

$$= \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

- **But this approach to determining the probability of a word sequence gets to be computationally very expensive <u>and</u> suffers from sparse data**

Markov Assumption (basic idea)

- How do we (efficiently) compute $P(w_n|w_1^{n-1})$?
- Trick (!): Instead of P(rabbit|I saw a), we use P(rabbit|a).
  - This lets us collect statistics in practice via a bigram model: P(the barking dog) = P(the|<start>) × P(barking|the) × P(dog|barking)

Markov Assumption (the very idea)

- Markov models are the class of probabil-istic language models that <u>assume</u> that we can predict the probability of some future unit *without looking too far* into the past
  - Specifically, for N=2 (bigram):
  - $P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$; w_0 := <start>
- Order of a Markov model: length of prior context
  - bigram is first order, trigram is second order, ...

Statistical HMM-based Tagging

[Brants, 2000]

- *State transition probability*: Likelihood of a tag immediately following n other tags
  - $P_1(Tag_i \mid Tag_{i-1} \ldots Tag_{i-n})$
- *State emission probability*: Likelihood of a word given a tag
  - $P_2(Word_i \mid Tag_i)$

    - die/DET Frau/NOUN ,/COMMA die/DET or PREL singt/VFIN

Trigrams for Tagging

- *State transition probabilities (trigrams)*:
  - P_1(DET | COMMA NOUN) = 0.0007
  - P_1(PREL | COMMA NOUN) = 0.0
- *State emission probabilities*:
  - P_2(die | DET) = 0.7
  - P_2(die | PREL) = 0.2
- Compute probabilistic evidence for the tag being
  - DET: $P_1 \cdot P_2 = 0.0007 \cdot 0.7 = 0.00049$
  - PREL: $P_1 \cdot P_2 = 0.01 \cdot 0.2 = 0.002$

Taken from (POS-annotated) corpora

- die/DET Frau/NOUN ,/COMMA die/PREL singt/VFIN

Inside (most) POS Taggers

- Lexicon look-up routines
- Morphological processing (not only deflection!)
- Unknown word handler, if lexicon look-up fails (based on statistical information)
- Ambiguity ranking (priority selection)

Chunking

Arginine methylation of STAT1 modulates IFN induced transcription

Chunking

[Arginine methylation] of [STAT1] modulates [IFN induced transcription]

Shallow Parsing

[Arginine methylation of STAT1]$_{NP}$ [modulates]$_{VP}$ [IFN induced transcription]$_{NP}$

Shallow Parsing

[[Arginine methylation]$_{NP}$ [of STAT1]$_{PP}$]$_{NP}$

[Arginine methylation of STAT1]$_{NP}$ [modulates]$_{VP}$ [IFN induced transcription]$_{NP}$

Shallow Parsing

[[IFN induced]$_{AP}$ [transcription]$_{N}$]$_{NP}$

[[Arginine methylation]$_{NP}$ [of STAT1]$_{PP}$]$_{NP}$

[Arginine methylation of STAT1]$_{NP}$ [modulates]$_{VP}$ [IFN induced transcription]$_{NP}$

Deep Parsing

[[IFN induced]$_{AP}$ [transcription]$_{N}$]$_{NP}$

[[[Arginine]$_{N}$ [methylation]$_{N}$]$_{NP}$ [[of]$_{P}$ [STAT1]$_{N}$]$_{PP}$]$_{NP}$

[[Arginine methylation]$_{NP}$ [of STAT1]$_{PP}$]$_{NP}$

[Arginine methylation of STAT1]$_{NP}$ [[modulates]$_{V}$ [IFN induced transcription]$_{NP}$]$_{VP}$

Deep Parsing

[[[IFN]$_N$ [induced]$_A$]$_{AP}$ [transcription]$_N$]$_{NP}$

[[IFN induced]$_{AP}$ [transcription]$_N$]$_{NP}$

[[[Arginine]$_N$ [methylation]$_N$]$_{NP}$ [[of]$_P$ [STAT1]$_N$]$_{PP}$]$_{NP}$

[[Arginine methylation]$_{NP}$ [of STAT1]$_{PP}$]$_{NP}$

[Arginine methylation of STAT1]$_{NP}$ [[modulates]$_V$ [IFN induced transcription]$_{NP}$]$_{VP}$

Chunking Principles

- Goal: divide a sentence into a sequence of chunks (ako phrases)
- Chunks are non-overlapping regions of a text
  - *[I] saw [a tall man] in [the park]*
- Chunks are non-exhaustive
  - not all words of a sentence are included in chunks
- Chunks are non-recursive
  - a chunk does not contain other chunks
- Chunks are mostly base NP chunks

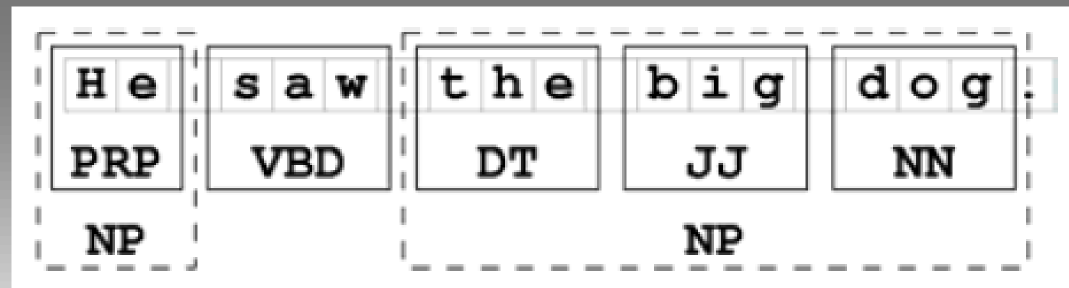
[[*the synthesis*]$_{\text{NP-base}}$ *of* [*long enhancer transcripts*]$_{\text{NP-base}}$]$_{\text{NP-complex}}$

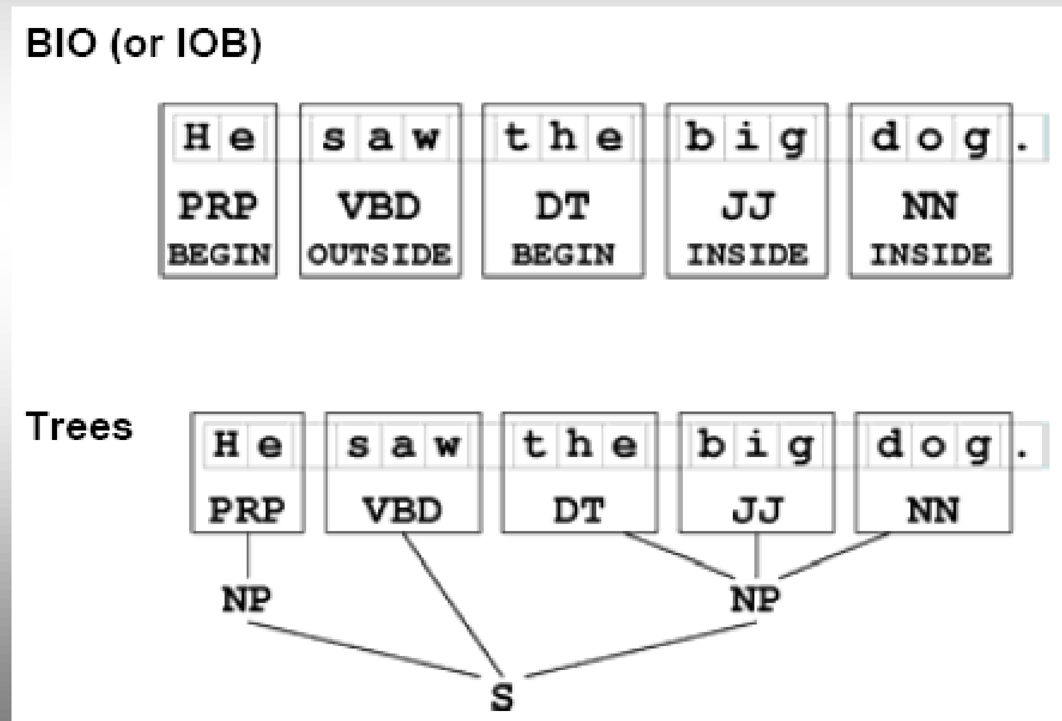
The Shallow Syntax Pipeline

Tagging

Chunking

Parsing





BIO Format for Base NPs

a	DT	B
mechanism	NN	I
that	WDT	B
increases	VBZ	O
NF-kappa	NN	B
B/I	NN	I
kappa	NN	I
B	NN	I
dissociation	NN	I
without	IN	O
affecting	VBG	O
the	DT	B
NF-kappa	NN	I
B	NN	I
translocation	NN	I
step	NN	I

A Simple Chunking Technique

- Simple chunkers usually ignore lexical content
  - Only need to look at part-of-speech tags
- Basic steps in chunking
  - Chunking / Unchunking
  - Chinking
  - Merging / Splitting

Regular Expression Basics

- “|” OR operator (explicit OR-ing)
  - “[a|e|i|o|u]” matches any occurrence of vowels
- “[abc]” matches any occurrence of either “a”, “b” or “c” (implicit OR-ing)
  - “gr[ae]y” matches “grey” or “gray” (but not “graey”)
- “.” matches arbitrary char
  - “d.g” matches “dag”, “dig”, “dog”, “dkg” …
- “?” preceding expression/char may or may not occur
  - “colou?r” matches “colour” and “color”
- “+” preceding expression occurs at least one time
  - “(ab)+” matches “ab”, “abab”, “ababab”, …
- “*” preceding expression occurs null time or arbitrary often
  - “(ab)*” matches “_”, “ab”, “abab”, “ababab”, …

Chunking

- Define a regular expression that matches the sequences of tags in a chunk
  - <DT>? <JJ>* <NN.?>
- Chunk all matching subsequences
  - *A*/DT *red*/JJ *car*/NN *ran*/VBD *on*/IN *the*/DT *street*/NN
  - [*A*/DT *red*/JJ *car*/NN] *ran*/VBD *on*/IN [*the*/DT *street*/NN]
- If matching subsequences overlap, the first one gets priority
- Unchunking is the opposite of chunking

Chinking

- A chink is a subsequence of the text that is not a chunk
- Define a regular expression that matches the sequences of tags in a chink
  - (<VB.?> | <IN>)+
- Chunk anything that is *not* a matching sub-sequence
  - *A*/DT *red*/JJ *car*/NN *ran*/VBD *on*/IN *the*/DT *street*/NN
  - [*A*/DT *red*/JJ *car*/NN] *ran*/VBD *on*/IN [*the*/DT *street*/NN]

chink

Merging

- Combine adjacent chunks into a single chunk
- Define a regular expression that matches the sequences of tags on both sides of the point to be merged
  - Merge a chunk ending in “JJ” with a chunk starting with “NN”, i.e. left: <JJ>, right: <NN.>
- Chunk all matching subsequences
  - [*A*/DT *red*/JJ] [*car*/NN] *ran*/VBD *on*/IN *the*/DT *street*/NN
  - [*A*/DT *red*/JJ *car*/NN] *ran*/VBD *on*/IN *the*/DT *street*/NN
- Splitting is the opposite of merging

Concluding Remarks

- Chunking – as the weakest form of syntactic structuring – relies on RegExs
- RegExs (formally) belong to the class of regular grammars
- Regular grammars and their (finite-state) automata have linear run-time complexity
- Standard CF grammars and their associated push-down automata have (at best) cubic run-time complexity
- Hence, there is a trade-off between different levels of richness of syntactic structures and gains/losses of run-time behavior

What are Named Entities?

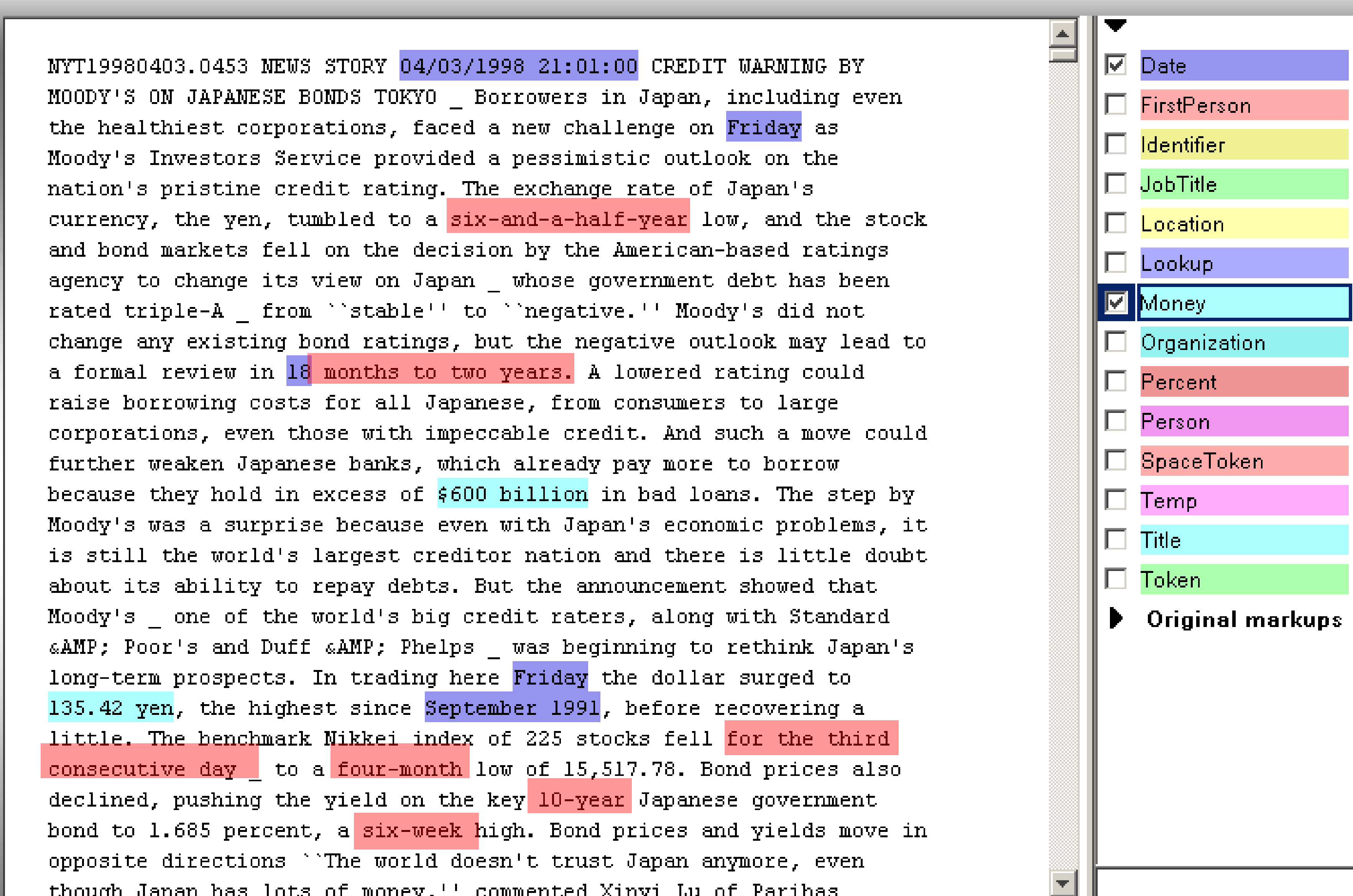
- Names of persons
  - *Dr. Jonathan Peeko, Professor Johnson*
- Names of companies or organizations
  - *Sony, United Nations, Texas Instruments, General Motors*
- Names of locations
  - *Paris, San Francisco, Rocky Mountains, Yellowstone Park*
- Date and time expressions
  - *Feb 17, 1973; 4.40p.m.; 16.40 Uhr; autumn 2000; last year*
- Addresses
  - *7 Ugly Way, Wolverhampton UH0 1Q5*
  - udo.hahn@uni-jena.de
- Names of proteins or genes or diseases,
  - *chloramphenicol acetyltransferase, NF-kappa B, SARS*
- Measure expressions
  - 420 kp, 21 l/m^2, 37%, 900€

What are Named Entities?

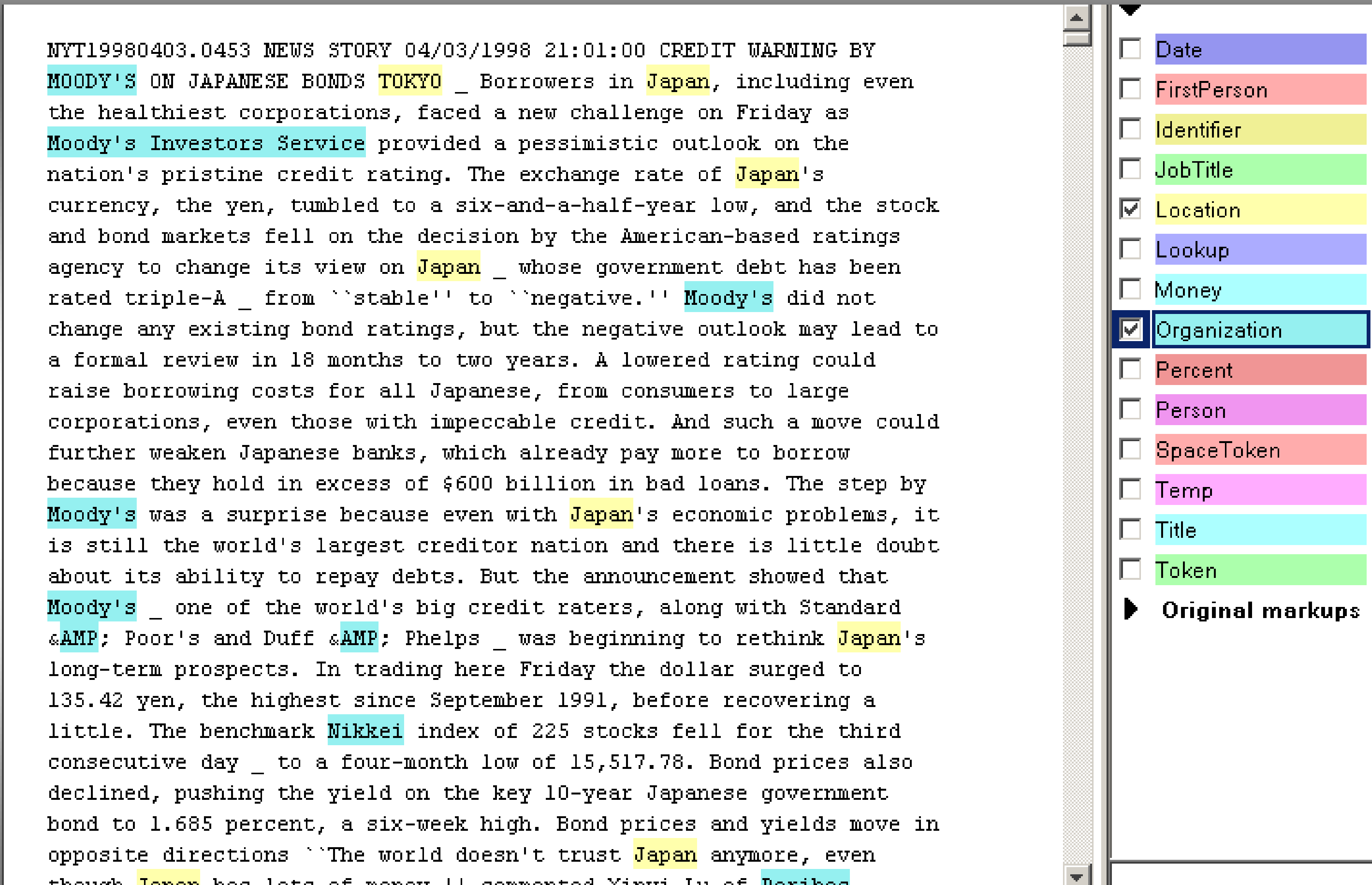
- Names of persons
  - *Dr. Jonathan Peeko, Professor Johns*
- Names of companies or organizations
  - *Sony, Uni... al Motors*
- Names of l...
  - *P... e Park*
- Date a...
  - *... year*
- Addre...
  - *... pton ... 40 1Q5*
  - udo.hah... ni-jena.de
- Names of proteins or genes or diseases,
  - *chloramphenicol acetyltransferase, NF-kappa B, SARS*
- Measure expressions
  - 420 kp, 21 l/m², 37%, 900€

named entities are intentionally <u>excluded from</u> the lexicon

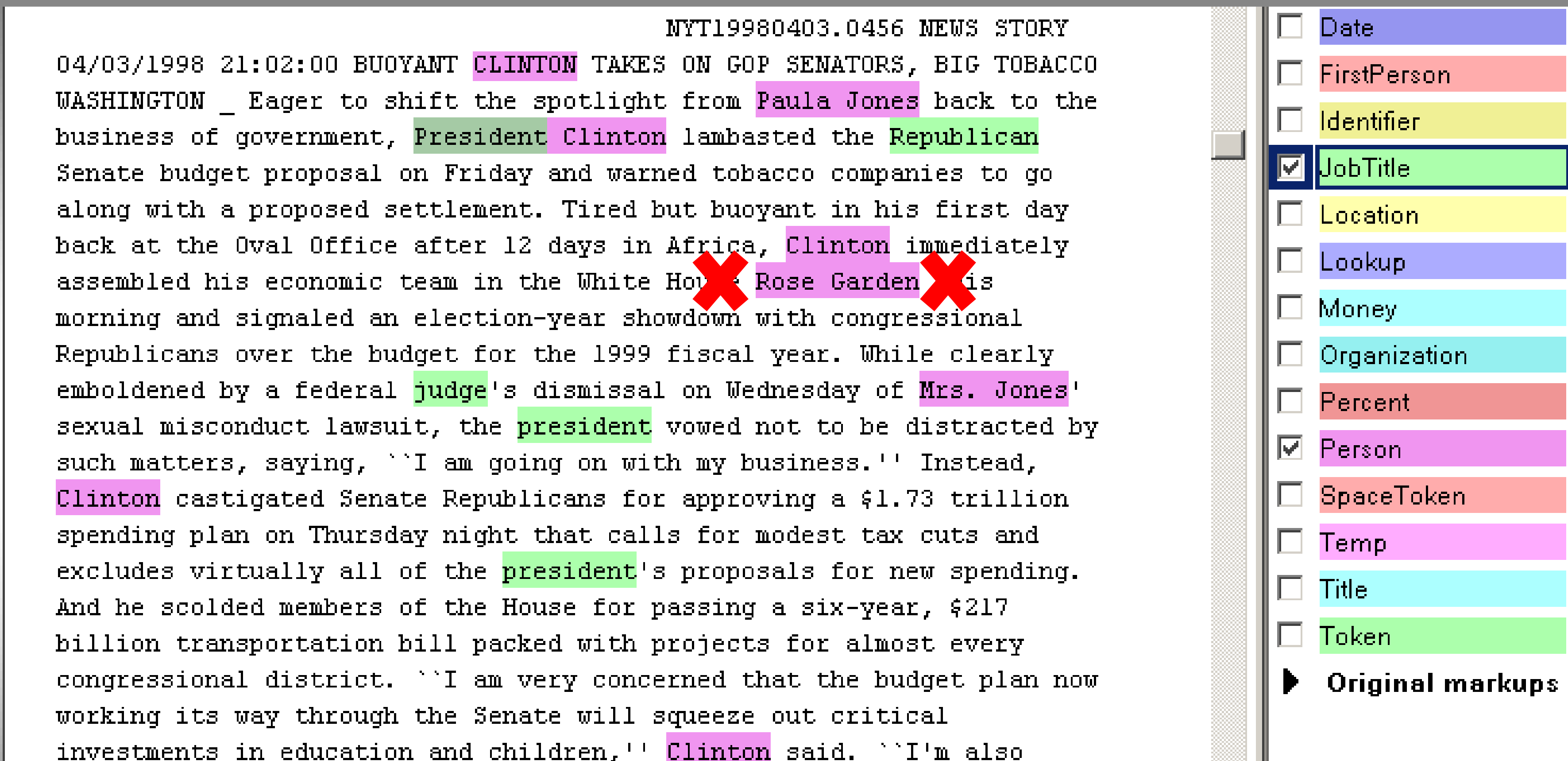
GATE: NER – Examples (1/3)



GATE: NER – Examples (2/3)



GATE: NER – Examples (3/3)



Two Types of NER Methods

Human Knowledge Engineering (symbolic p.)

- rule based
- developed by experienced language engineers
- based on human intuition
- requires only small amount of plain training data
- development can be very time consuming
- some changes may be hard to accommodate

(Supervised) Machine Learning Systems (empir.p.)

- use statistics or other machine learning technique
- developers do (almost) not need linguistic expertise
- fully automatic
- requires large amounts of annotated training data
- annotators are cheap (but you get what you pay for!)
- some changes may require re-annotation of the entire training corpus

Naïve NER Method: List Look-up

- Recognize entities stored in given lists
  - *gazetteers*, e.g., online phone directories, yellow pages)
- Advantages:
  - simple, fast, language independent, easy to retarget (just create lists)
- Disadvantages:
  - impossible to enumerate all names and name variants, collection and maintenance of lists

NER by Pattern Recognition

- **Names often have internal structure - these components can be either stored or guessed, e.g., for "Location" we have RegEx-style constraints such as:**

Capitalized Word + {City, Forest, Center, River}

which yields: ***Sherwood Forest, Manchester City, Rhine River***

Capitalized Word + {Street, Boulevard, Avenue, Road}

which yields: ***Portobello Street, Washington Avenue***

NER by Expressive Rules

- Context-sensitive rules of the kind:

$$A \rightarrow B \setminus C / D$$

  - A is a set of attribute-value expressions and optional score, the attributes refer to elements of the input token feature vector
  - B, C, D are sequences of attribute-value pairs and regular expressions; variables are also supported
  - B and D are left and right context, respectively, and can be empty (hint: read backwards!)

Example: [syn=NP, sem=ORG] (0.9) →
\ [norm="university"], [token="of"],
[sem=REGION|COUNTRY|CITY] / ;

NER by Machine Learning

- NE task is frequently broken down in two parts:
  - Recognizing the entity boundaries
  - Classifying the entities in the NE categories
- Features are at least as important as the choice of the ML method
  - Simple pattern matching of orthographic features: capitalization, punctuation marks, numerical symbols
  - Windows for lexical features (e.g., “Mr.” for persons)
  - Affix features (“-ase” for proteins, “”-ectomy” for medical procedures, etc.”)
  - POS info (and chunks)

Merkmale für die Zuordnung von Named Entities

Feature	Explanation
Lexical items	The token to be labeled
Stemmed lexical items	Stemmed version of the target token
Shape	The orthographic pattern of the target word
Character affixes	Character-level affixes of the target and surrounding words
Part of speech	Part of speech of the word
Syntactic chunk labels	Base-phrase chunk label
Gazetteer or name list	Presence of the word in one or more named entity lists
Predictive token(s)	Presence of predictive words in surrounding text
Bag of words/Bag of N-grams	Words and/or *N*-grams occurring in the surrounding context

Shape	Example
Lower	cummings
Capitalized	Washington
All caps	IRA
Mixed case	eBay
Capitalized character with period	H.
Ends in digit	A9
Contains hyphen	H-P

Features for Machine Learning (CoNLL 2003 Shared Task)

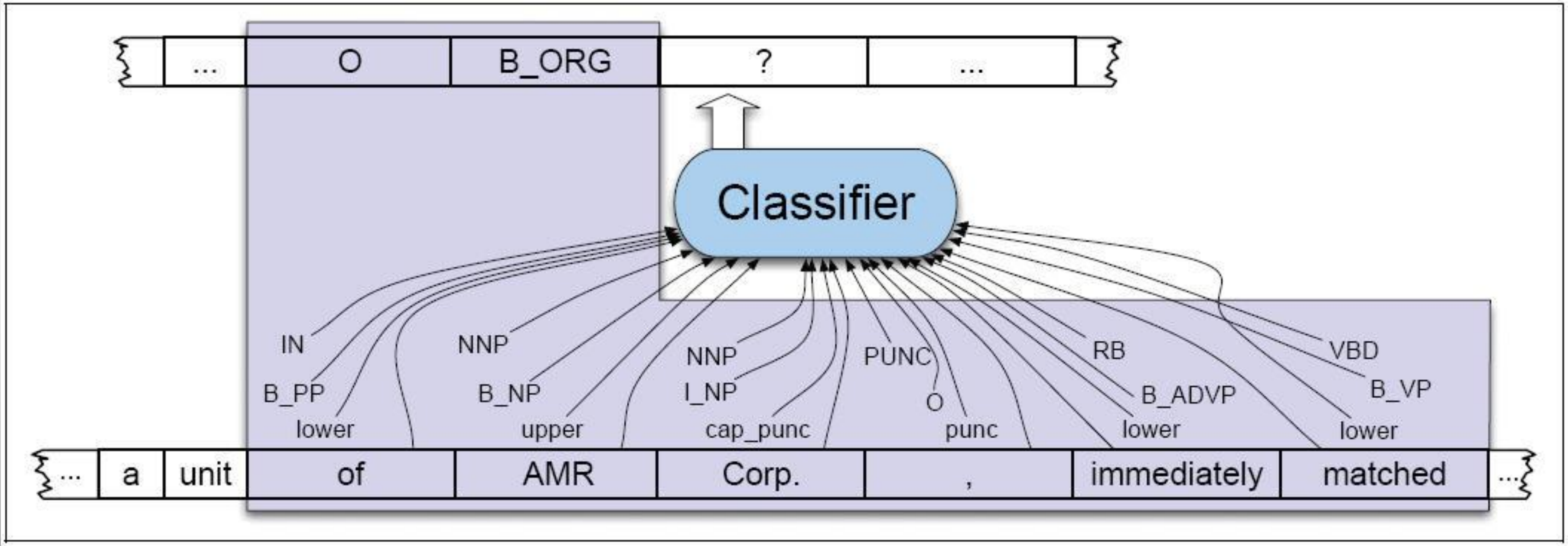
	lex	pos	aff	pre	ort	gaz	chu	pat	cas	tri	bag	quo	doc
Florian	+	+	+	+	+	+	+	-	+	-	-	-	-
Chieu	+	+	+	+	+	+	-	-	-	+	-	+	+
Klein	+	+	+	+	-	-	-	-	-	-	-	-	-
Zhang	+	+	+	+	+	+	+	-	-	+	-	-	-
Carreras (a)	+	+	+	+	+	+	+	+	-	+	+	-	-
Curran	+	+	+	+	+	+	-	+	+	-	-	-	-
Mayfield	+	+	+	+	+	-	+	+	-	-	-	+	-
Carreras (b)	+	+	+	+	+	-	-	+	-	-	-	-	-
McCallum	+	-	-	-	+	+	-	+	-	-	-	-	-
Bender	+	+	-	+	+	+	+	-	-	-	-	-	-
Munro	+	+	+	-	-	-	+	-	+	+	+	-	-
Wu	+	+	+	+	+	+	-	-	-	-	-	-	-
Whitelaw	-	-	+	+	-	-	-	-	+	-	-	-	-
Hendrickx	+	+	+	+	+	+	+	-	-	-	-	-	-
De Meulder	+	+	+	-	+	+	+	-	+	-	-	-	-
Hammerton	+	+	-	-	-	+	+	-	-	-	-	-	-

Table 3: Main features used by the the sixteen systems that participated in the CoNLL-2003 shared task sorted by performance on the English test data. Aff: affix information (n-grams); bag: bag of words; cas: global case information; chu: chunk tags; doc: global document information; gaz: gazetteers; lex: lexical features; ort: orthographic information; pat: orthographic patterns (like Aa0); pos: part-of-speech tags; pre: previously predicted NE tags; quo: flag signing that the word is between quotes; tri: trigger words.

Merkmalskodierung für NEs

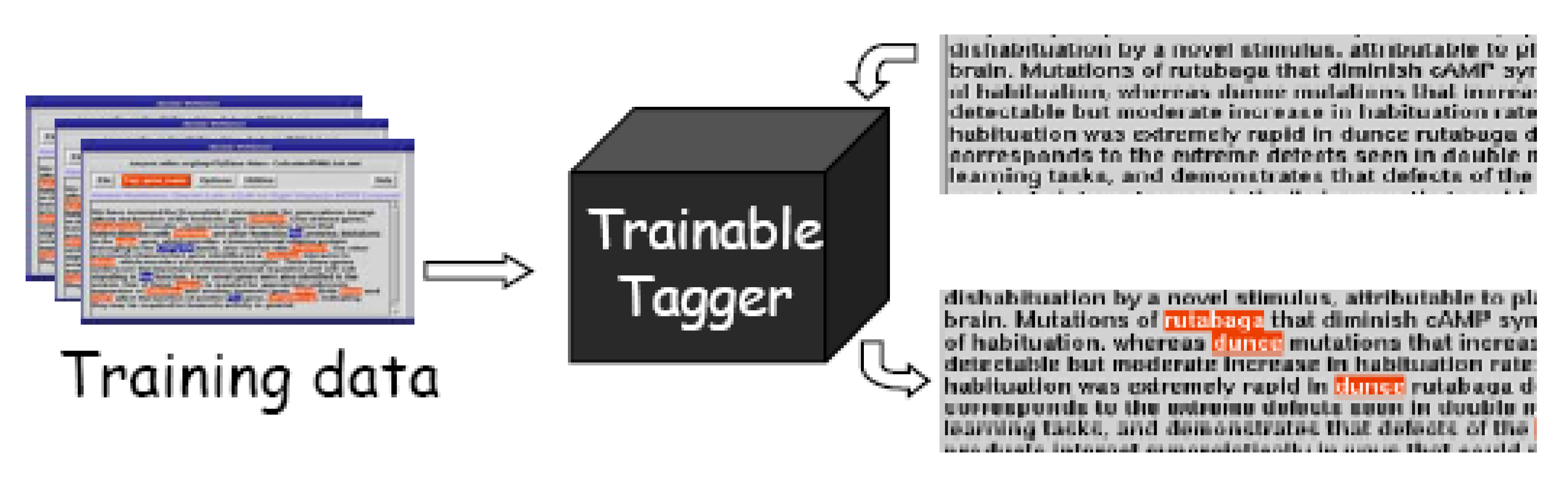
Features				Label
American	NNP	B_{NP}	cap	B_{ORG}
Airlines	NNPS	I_{NP}	cap	I_{ORG}
,	PUNC	O	punc	O
a	DT	B_{NP}	lower	O
unit	NN	I_{NP}	lower	O
of	IN	B_{PP}	lower	O
AMR	NNP	B_{NP}	upper	B_{ORG}
Corp.	NNP	I_{NP}	cap_punc	I_{ORG}
,	PUNC	O	punc	O
immediately	RB	B_{ADVP}	lower	O
matched	VBD	B_{VP}	lower	O
the	DT	B_{NP}	lower	O
move	NN	I_{NP}	lower	O
,	PUNC	O	punc	O
spokesman	NN	B_{NP}	lower	O
Tim	NNP	I_{NP}	cap	B_{PER}
Wagner	NNP	I_{NP}	cap	I_{PER}
said	VBD	B_{VP}	lower	O
.	PUNC	O	punc	O

Named Entity Tagging als Sequence Labeling-Problem



Systemarchitektur für (überwachtes) Maschinelles Lernen

Merkmale = beobachtbare Indikatoren (in den Trainingsdaten)	**Algorithmen für Maschinelles Lernen** = Rechenverfahren zur Bestimmung von (statistischen) Modellen über die Verteilung von Merkmalen (in den Trainingsdaten)



Algorithmen für (überwachtes) Maschinelles Lernen [Flach 2012, Murphy 2012]

- Einfache Klassifikatoren (Classifier)
  - Naive-Bayes´scher Klassifikator
  - k-Nächster Nachbar (k-nearest neighbor)
  - Entscheidungsbäume (decision trees)
- Hochdimensionale Klassifikatoren (Classifier)
  - Support Vector Machines (SVM)
- (strukturorientierte) Graphische Modelle
  - Hidden-Markov-Modelle
  - Conditional Random Fields (CRF)
  - Bayes´sche Netze
- (Künstliche) neuronale Netze ⇨ Deep Learning
- Genetische Algorithmen

Machine Learning–General Task

A computer program is said to *learn*

- from experience **E** (data in the form of represent-ative examples / instances of the whole input space)
- with respect to some class of tasks **T**
- and performance measure **P**,
- if its performance at tasks **T** as measured by **P**, improves with experience **E**

- Learned hypothesis: model of problem/task **T**
- Model quality: accuracy/performance measured by **P**

Machine Learning – Two Fundamental Modes

- Supervised learning
  - Given : Training examples (training set T)

    $\{ (x_1, f(x_1)) , (x_2 , f(x_2)), \ldots (x_n, f(x_n)) \}$

    for some unknown function y = f (x)
  - Find : f (x)
  - Predict y' = f (x') where x' is not in the training set but T-wise similar data sets
- Unsupervised learning
  - Given : data (data set D)

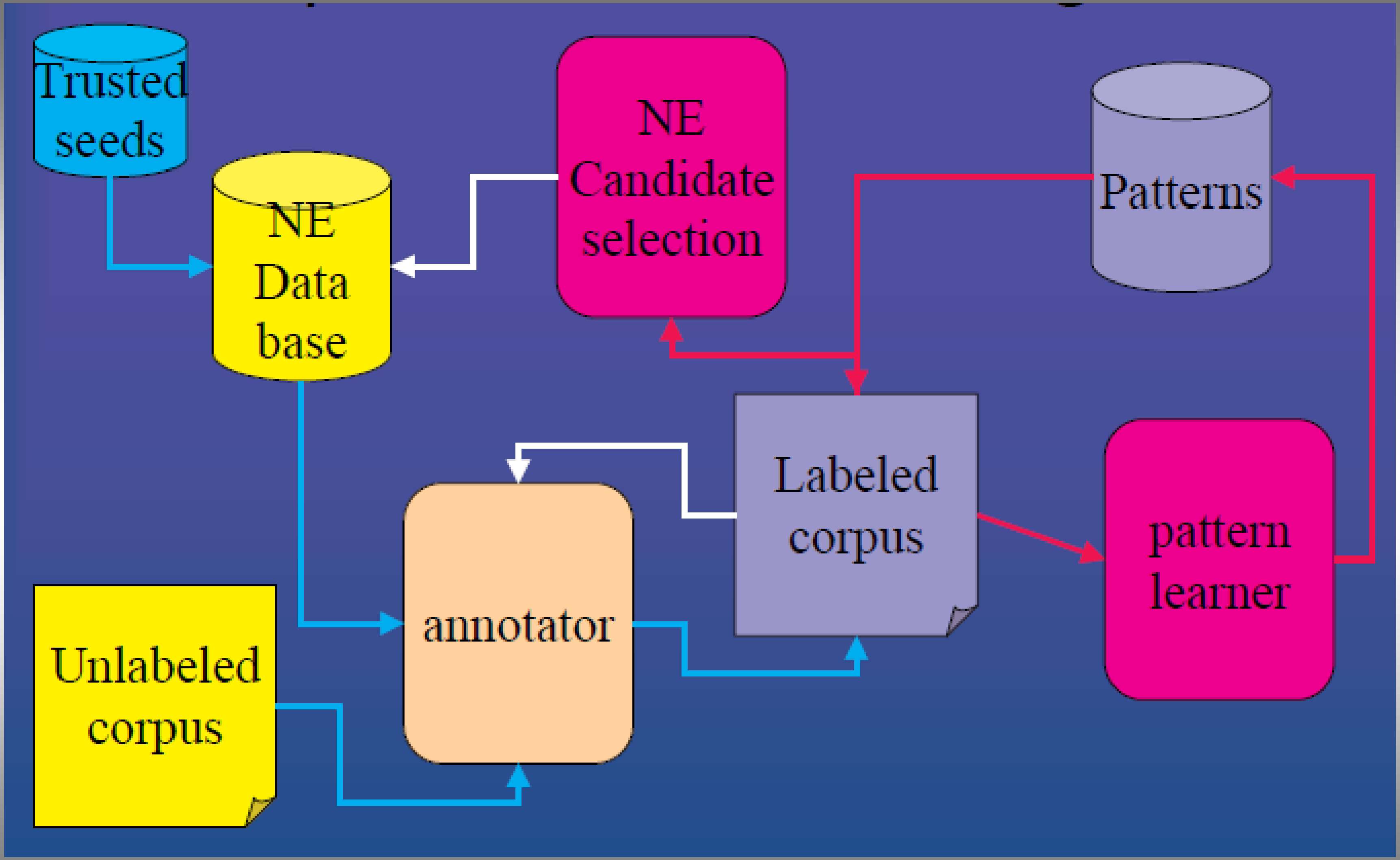
    $\{ x_1 , x_2 , \ldots, x_n \}$

    for some unknown function y = f (x)
  - Find : f (x)
  - Predict y = f (x) where x is in the data set or D-wise similar data sets

Basic Idea for (Almost) Unsupervised NER

- Define manually only a small set of trusted seeds (a bit of ground truth)
- Training then only uses unlabeled data
- Initialize system by labeling the corpus with the seeds
- Extract and generalize patterns from the context of the seeds
- Use the patterns to further label the corpus and to extend the seed set (*bootstrapping*)
- Repeat the process unless no new terms can be identified

Architecture for (Almost) Unsupervised NER



Learning Ordered Decision Rules

- The task: to learn a decision list to classify strings as person, location or organization

The learned decision list is an *ordered* sequence of if-then rules

… *says Mr. Gates, founder of Microsoft* …

R_1 : if features then person
R_2 : if features then location
R_3 : if features then organization
…
R_n : if features then person

… *says Mr. Gates, founder of Microsoft* …

Outline of Unsupervised Co-Training

- **Parse an unlabeled document set → syntactic units**
- **Extract each NP whose head is tagged as Proper Noun (Proper Noun is supertype of NEs: NER as sub-typing)**
- **Define a set of relevant features which can be applied to extracted NPs**
- **Define two separate types of rules on the basis of the feature space**
- **Determine small initial set of seed rules**
- **Iteratively extend the rules through co-training**

Two Types of Rules

- **Spelling Rules**
  - **Rules which directly specify lexical conditions (e.g., “Mr.” ⇨PERSON)**
- **Contextual Rules**
  - **Rules which specify co-occurring lexical or phrasal conditions (e.g., “president” co-occurs with “Mr.” ⇨PERSON)**

- **N.B.: Huge amount of unlabeled data in a corpus gives useful hints!**

Kinds of Noun Phrases and Spelling-Context Pairs

1. There was an appositive modifier to the NP, whose head is a singular noun (tagged NN).
   - *…says [Maury Cooper], [a vice president]…*

2. The NP is a complement to a preposition which is the head of a PP. This PP modifies another NP whose head is a singular noun.
   - *… fraud related to work on [a federally funded sewage plant] [in [Georgia]].*

- *…says Maury Cooper, a vice president…*
  - (Maury Cooper, president)

- *… fraud related to work on a federally funded sewage plant in Georgia.*
  - *(Georgia, plant_in)*

Features

- Set of spelling features
  - Full-string=x (full-string=Maury Cooper)
  - Contains(x) (contains(Maury))
  - Allcap1 IBM
  - Allcap2 N.Y.
  - Nonalpha=x A.T.&T. (nonalpha=..&.)
- Set of context features
  - Context = x (context = president)
  - Context-type = x appos or prep

Examples of Features

Sentence	Entities(Spelling/Context)	(Active) Features
But Robert Jordan, a partner at Steptoe & Johnson who took …	Robert Jordon/partner	Full-string=Robert_Jordan, contains(Robert), contains(Jordan), context=partner, context-type=appos
	Steptoe & Johnson/partner_at	Full-string=Steptoe_&_Johnson, contains(Steptoe), contains(&), contains(Johnson), nonalpha=& , context=partner_at, context-type=prep
By hiring a company like A.T.&T. …	A.T.&T./company_like	Full-string= A.T.&T., allcap2, nonalpha=..&. , context=company_like, context-type=prep
Hanson acquired Kidde Incorporated, parent of Kidde Credit, for …	Kidde Incorporated/parent	Full-string=Kidde_Incorporated, contains(Kidde), contains(Incorporated), context=parent, context-type=appos
	Kidde Credit/parent_of	Full-string=Kidde_Credit, contains(Kidde), contains(Credit), context=parent_of, context-type=prep

Formal Structure of Rules

Rules

Two separate types of rules:
Spelling rules
Context rules

Feature → NE-type, h(Feature,NE-type)

h(x,y): the strength of a rule, defined as

Count(x, y) **is the number of times feature x is seen with label y in training data,**

$$\arg\max_{x,y} \frac{Count(x,y)+\alpha}{Count(x)+k\alpha}$$

where

$Count(x) = \sum_{y \in Y} Count(x,y)$

α is a smoothing parameter

$k = \#NE\text{-}types$

Is an estimate of the conditional probability of the NE-type given the feature, P(y|x)

The rules ordered according to their strengths h form a decision list: the sequence of rules are tested in order, and the answer to the ***first*** satisfied rule is output.

7 Seed Rules

7 SEED RULES

- Full-string = New York → Location
- Full-string = California → Location
- Full-string = U.S. → Location
- Contains(Mr.) → Person
- Contains(Incorporated) → Organization
- Full-string=Microsoft → Organization
- Full-string=I.B.M. → Organization

Note: only one type of rules used as seed rules, and all NE-types should be covered

Co-Training Algorithm

1. Set N=5 (max. # of rules of each type induced in each iteration)
2. **Initialize**: Set the spelling decision list equal to the set of seed rules. Label the training set using these rules.
3. Use these to get contextual rules. (x = feature, y = label)
    1. Compute h(x,y), and induce at most N * K rules
    2. all must be above some threshold p_{min}=0.95

K = # NE types

4. Label the training set using the contextual rules.
5. Use these to get N*K spelling rules (same as step 3.)
6. Set spelling rules to seed plus the new rules.
7. If N < 2500, set N=N+5, and goto step 3.

8. Label the training data with the combined spelling/contextual decision list, then induce a final decision list from the labeled examples where all rules (regardless of strength) are added to the decision list.

Example

- (IBM, company)
  - …IBM, the company that makes…
- (General Electric, company)
  - ..General Electric, a leading company in the area,…
- (General Electric, employer)
  - … joined General Electric, the biggest employer…
- (NYU, employer)
  - NYU, the employer of the famous Ralph Grishman,…

Power of the Algorithm

- Greedy method
  - At each iteration method increases number of rules
  - While maintaining a high level of agreement between spelling & context rules

For n= 2500:

1. The two classifiers give both labels on 49.2% of the unlabeled data
2. And give the *same* label on 99.25% of these cases

➢ The algorithm maximizes the number of unlabeled examples on which the two decision list agree.

Evaluation of the Algorithm

- 88,962 (spelling, context) pairs.
  - 971,746 sentences
- 1,000 randomly extracted to be test set.
- Location, person, organization, noise (items outside the other three)
- 186, 289, 402, 123 (- 38 temporal noise).
- Let N_c be the number of correctly classified examples
  - Noise Accuracy: N_c / 962

Results

Algorithm	Clean Accuracy	
Baseline	45.8%	
EM	83.1%	
Yarowsky 95	81.3%	
Yarowsky Cautious	91.2%	
DL-CoTrain	91.3%	
CoBoost	91.1%	

Remarks

- Needs full parsing of unlabeled documents
  - Restricted language independency
  - Need linguistic sophistication for new types of NE
- Slow training
  - In each iteration, full size of training corpus has to be re-labeled

Resources for NLP

- Empirical (Learning) Paradigm for NLP
- Types of Resources
  - Language data (plain, annotated)
  - Systems for acquiring and maintaining language data
  - Computational lexicons and ontologies
  - NLP Core Engines
  - NLP Application Systems
  - Machine Learning Resources
- Methodological Issues of NLP Resources

Ressourcen für die Sprachverarbeitung

- Referenzkorpora (Nationalkorpora)
  - Standardsprache (Zeitungen, Belletristik)
- Non-Standard-Korpora
  - Informelle Sprache (Chats, Blogs, E-Mails)
  - Fachsprachen (z.B.: klinische Berichte)
- Rohdaten vs. Annotation
  - Linguistische Metadaten
    - Morphologie, Syntax, Semantik, Pragmatik

Language Data

- Plain language data
  - Just text or speech
    - ASCII/UTF-8-compatible, pdf, HTML/SGML
- Annotated language data
  - Enriched by linguistic meta-data
    - Linguistic annotation languages (XML)