Informationsmanagement

Σ Alphabet = Menge von Symbolen/Zeichen

$s = s_1 \dots s_n \in \Sigma^*$ Zeichenkette

Daten - ausgetauschte Nachrichten
└ Zeichenketten nach Grammatik gebildet

Problem-lösung:
- Interpretation
- **Informationen** + Interpretationsbezug (Semantik) für Empfänger
- Wissenserwerb
- **Wissen** - Gesamtheit Kenntnisse, Fähigkeiten zur Problemlösung + Vernetzung zu bekannten Infos

Strukturierungsgrad

strukturierte Daten
gleichartige Struktur, Datenmodell folgend
z.B. Listen, Tabellen, Diagramme, Java Objekte

semistrukturierte Daten
ohne festes Datenmodell
aber Daten implizieren Struktur oder erweiterbares Datenmodell
z.B. XML-Dokument
Object Exchange Model

„unstrukturierte Daten"
ohne formalisierte (inhaltliche) Struktur
z.B. natürlichsprachlicher Text
Bild, Musik (schwer für Computer)
(Text Mining?, Natural Language Processing)

Inf. Management

Daten organisieren, strukturieren
speichern, abfragen, löschen, ändern
pflegen,
interpretieren, vernetzen
analysieren, verifizieren

Inf. System Anforderungen

Informationsenthalt
nötige Daten ableitbar

Konsistenzerhaltung
nur vernünftige Daten speichern
möglichst Redundanzfrei
Platz sparen, Anomalien vermeiden

Entwurfsprozess

1. Anforderungsanalyse
Fachwissen, Terminologie, Inf.-bedarf
⇒ Informelle Dokumentation z.B. Interviews, Texte

2. Konzeptioneller Entwurf / design
Abstrakte Modellierung Domäne
Objekte, Beziehungen?
⇒ formale Beschreibung z.B. ER-Modell
Welche Daten? UML-Strukturm.

3. Verteilungsentwurf / distributed system design
Fragmentieren Daten,
Sync & Replica

4. Logischer Entwurf impl. spezifisch, Geräte unabhängig
konzeptionelle Modelle → Konzepte eingesetztes Inf.-System
⇒ Logische Datenmodelle z.B. relationales Modell
Wie?

5. Datendefinition
Deklaration / Programmierung Datenmodel in Inf.-System
⇒ Definitionssprachen z.B. SQL

6. Physischer Entwurf
Zugriffs-, Speicherstrukturen auf disk

7. Implementierung & Wartung
Interne Verarbeitung Anfragen
Anfrageoptimierung

BRUNNEN Themenüberblick auf slides

Relationale DB's

Backend Unternehmensdaten
Web-Bereich
Android, Mac OS, SQLite, ...
SQL wichtiger Skill (auch Python, R)

(2) Konzeptionelle Datenmodellierung (Externe Ebene DB)

Anforderungen → konzep. Datenmodell

Konzeptionelle Datenmodellierung

Objekte, Merkmale, Beziehungen, Regeln?

Entity-Relationship (ER) Modelle

statische Eigenschaften (keine Funktionen/Methoden)

beliebt => „Laien"-verständlich, intuitiv

graphische Notation nach Chen

Entitätstyp

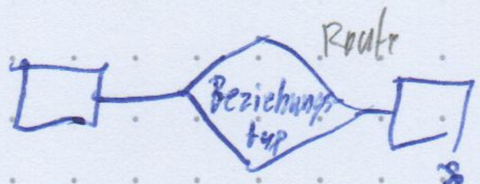
Menge Entitäten gleichen Typs
(individuell identifizierbare Objekte, Instanzen, ∈ Typ)

Entitätstyp — Rechteck

Entitätstyp (Attributes, ...)

Beziehungstyp

zwischen ≥2 Entitätstypen
Relation $\subseteq E_1 \times \dots E_n$
Beziehung konkret zwischen Entitäten



Beziehungstyp (E_1, E_2, ...; Attributes, ...)

Attribute / Merkmale

Ovd.

Schlüsselattribute

≥ Attribute Entitätstyp
identifiziert Entität eindeutig
Modellierungsentscheidung
ggf. künstlich z.B. ID

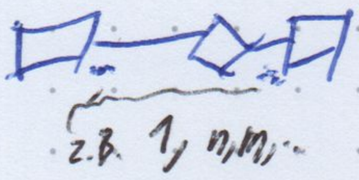
nur Entitätstypen

Unterstreichen Attribut

Rekursiver Beziehungstyp

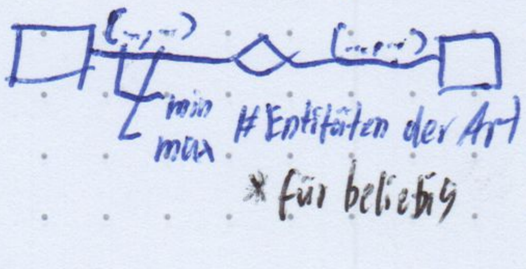
Funktionalitäten / Kardinalitäten für Beziehungstyp

#konkrete Beziehungen Entität max. beteiligt



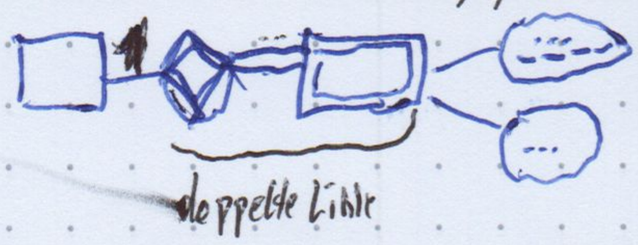
z.B. 1:1, 1:N, N:1, N:M

(min, max) - Notation präziser als Kardinalitäten



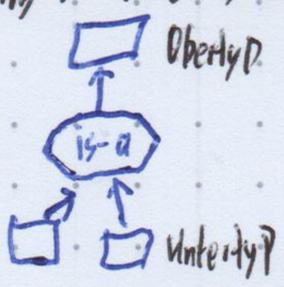
⚠ 1:N => (0,*) (1,1)

Schwache (existenzabhängige) Entitätstypen



nicht Klausur-relevant

Generalisierung / Vererbung der Attribute



Aggregation (Teil-Ganzes Beziehung)

Relationales Datenmodell beim logischen Entwurf

aus Relationen ≠ Tabelle
↳ keine Reihenfolge Attribute
Tupel

$D = \{D_1, ...\}$ Menge Datentypen/Domänen

$R = \{Attribute_1, A_2, ...\}$ Schema
$dom: R \to D$

$r(R) \subseteq dom(A_1) \times ... \times dom(A_n)$
Relation - Ausprägung von R

$t \in r$ Tupel
$t(A_i)$ Wert Attribut
$t(\alpha)$ Teiltupel $\alpha \subseteq R$

Schlüssel Attributmenge K (einfach $|K|=1$ / zusammengesetzt $|K|>1$ / trivial $K=R$

Superschlüssel
K schlüsseleindeutig $\Leftrightarrow \forall t_1, t_2 \in r: t_1(K) = t_2(K) \Rightarrow t_1 = t_2$

UI

Schlüsselkandidat / Schlüssel
K minimal $\Leftrightarrow \nexists K' \subseteq K: K'$ schlüsseleindeutig

$K_R = \{\{...\}, ...\}$

Primärschlüssel (PK)
1 gewählter Schlüsselkandidat

$PK_R = \{...\}$

Fremdschlüssel
$K' \subseteq$ Schema S bezüglich Schema R $\Leftrightarrow$ Referenz $p(K') = PK$ von R

$FK_R = \{D \to S.D\}$ (Attribut)
$\dots \to S.A \dots$

Referenzierter Wert existiert

1 Referenzielle Integrität
$\forall t_s \in S \; \exists t_r \in r: t_s(K') = t_r(p(K'))$

$R(\underline{A_n: INT}, B: INT, C: INT)$
↑ PK

FKs implizit durch Namensgleichheit

null Wert möglich

ER → Relationales Model

Regel 1
Entitätstyp ↦ Schema
↳ Schlüssel ⇒ PK falls minimal
sonst neu

Regel 2
Beziehungstyp ↦ Schema
$R(E_1, ...)$ $R(\underline{A_{11}, ... A_{1j}, ...}, \underline{A_{R1}, ... A_{Rm}})$
Fremdschlüssel E_1 Eigene Attr
↳ Benennung
- Name der referenzierten (falls eindeutig)
- künstlicher Name z.B. EntitätsnameAttributname
- Rollenname falls vorhanden

PKs alle Attr. von E. mit Kardinalität N_i falls keiner einer mit 1

Regel 3
Zusammenfassen Schemata mit gleichen PKs
Ausnahme: schlecht wenn Ergebnis dünn besetzt

Regel 4
Schwache Entitäten
Fremdschlüssel für starke
Teil PK

Relationale ~~Algebra~~ Abfragesprache

RWK - Relationenwertebereichkalkül
RTK - Relationentupelkalkül
RA - Relationenalgebra
) gleiche Aussagekraft

Codd's Algebra

Selection
$\sigma_p(S)$
- Konstantenselektion
- Attributselektion
- Logische Verknüpfung $\vee, \wedge, \neg$

Attribut θ $\{=, <, >, \leq, \geq\}$ Konstante / Attribut

Projektion
$\Pi_\alpha(S)$
$L \subseteq R$

Kartesisches Produkt $\times$

Erweiterungen

Joins
$\bowtie$ natürlich $\leftrightarrow$ $\bowtie_p$ allgemein
- vereinigt gleichnamige
- $R.\ldots = S.\ldots$
- Attr. bleiben „doppelt"

Join
$R \bowtie S$ = in beiden drinnen

Full outer Join
$R \bowtie L$ = fehlende Tuppel Attr = Null

Left "
~~$R \bowtie L$~~ $R \bowtie L$ = alle aus R

Right "
$R \bowtie L$ = alle aus L

Left / Right Semi-Join
$R \ltimes L = \Pi_R(R \bowtie L)$

Keine Duplikate

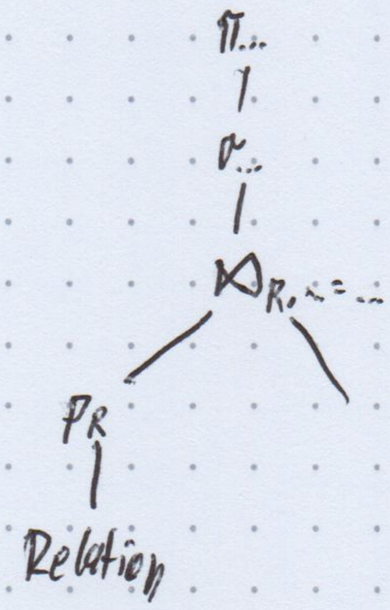
Mengenoperatoren
auf Tupelmengen

bei Vereinigungsverträglichkeit
- gleicher Grad = # Attribute
- gleiche Domänen

Vereinigung
$R \cup S$

Durchschnitt
$R \cap S = R - (R - S)$

Differenz
$R - S$



Relationale Division
Attribute / Projektion

$R \div S = \{t \in R-S \mid \forall s \in S : \exists r \in R : r(S) = s(S)\}$

$= \Pi_{(R-S)}(R) - \Pi_{(R-S)}((\Pi_{(R-S)}(R) \times S) - R)$

gruppieren nach R-S alle R-S zurückgeben für die Einträge für $\forall s \in S$ existieren

Umbenennung
Relation $\rho_{V1}(S) = V1$

Attribute ~~$\rho_{\text{Alt} \leftarrow A,\ldots}$~~ $A_{\text{Alt}} \leftarrow A, \ldots$

SQL - Structured English Query Language ↔ Relationale Algebra
auf Multimengen └imperativ ohne Duplikate
deklarative Anfragesprache für rel. DB

DQL - Data Query Language
DDL " Definition "
DML " Manipulation "
DCL " Control

DQL

select [distinct] * | Attr1, Attr2, ...

from Table1, ...

[natural join Table]*
[cross join Table on Table.Attr = Table.Attr.]*
[inner], (left (right | full) outer join

Reihenfolge kann durch Klammern geändert werden

[where Prädikat
  ├ kombinieren AND, OR, NOT
  ├ Vergleichop. =, <, <=, >, >=, ...
  ├ between ... and ...
  └ case (when ... then ...)* [else ...] end)

Pattern Matching Strings ... like ' ... '
  ├ _ beliebiges Zeichen
  └ % beliebige Zeichenkette auch ε

[group by Attr1, ...
 [having ...

// Aggregationsfunktionen
  count, sum, avg, max, min
// Relationale Algebra
  $\pi_{select}(\sigma_{having}(\chi_{groupby}(\sigma_{where}(from))))$

alle in select, ~~where~~ having, in group by oder Aggregate

]]

[order by Attr1 [asc | desc], ...]
  └ ending
  default asc.

[limit n] ; // MySQL, PostgreSQL; select top n : SQL Server; fetch first n rows only : Oracle

Umbenennung von Tabellen in from, join [as] newName
  Attr. in select [as] newName

Unterabfragen (select...) in Äußerem select, wenn nur ein Aggregat in select Unterabfrage als Wert
  └ innere
  from als Tabelle
  where als Wert
  als Tabelle
    ├ EXISTS T - T nicht leer
    ├ s IN T
    └ s op ALL T
      ANY

cast (value/attr as type)

Relationale Division select (R-S) from R group by (R-S) having count(*) = (select count(*) from S)

NULL-Werte
  dreiwertige Logik
  vergleichops → unknown ⇔ min 1 Operand Null
  and, or, not → unknown außer Kurzschlusslogik
  Aggregatfunc. ignorieren NULL
  Gruppierung NULL eigene Gruppe

zusätzliche Ops
  ... is NULL
  ... is not NULL

Recent Directions
  Data Lake - Tabellen ohne Schema darauf automated Join Detection
  NL-TO-SQL Natural Language —LLM→ SQL statt

BRUNNEN

Relationale Entwurfstheorie

Redundanzen
- Speicherplatzverschwendung

kann ⇒ ~~potentiell~~ Anomalien kann ⇒ Inkonsistenz
- Update nicht alle Stellen (Tupel, ~~Tabelle~~)
- Einfüge benötigt evtl. Dummy daten
- Lösch ~~beachte~~ z.B. Best. lösche Lieferanten mit, wenn in keinen anderen Best.

FD - functional dependencie

$\beta \subseteq R$ von $\alpha \subseteq R$ funktional abhängig (α Determinante für β)
$\Leftrightarrow \forall t_1, t_2 \in r(R): t_1(\alpha) = t_2(\alpha) \Rightarrow t_1(\beta) = t_2(\beta)$ α bestimmt β $\alpha \to \beta$

für jeden Wert von α genau ein Wert von β

voll FD $\Leftrightarrow \alpha \to \beta \wedge \nexists \alpha' \subset \alpha : \alpha' \to \beta$
partiell " $\exists$

β transitiv abhängig von α
$\Leftrightarrow \exists \gamma : \gamma$ voll FD von $\alpha \wedge \beta$ voll FD von γ
$\alpha \to \gamma \to \beta$

Attributhülle α^+ = Menge von α funk. abhängige Attribute

Algorithmus
$\alpha^+ = \alpha$
while Änderung
foreach FD $\beta \to \gamma$
if $\beta \subseteq \alpha^+$ then $\alpha^+ = \alpha^+ \cup \gamma$

Schlüsselkandidaten aus FDs

1. Attr. nicht auf rechter Seite FDs
   ⇒ muss in Schlüsselk. vorkommen
2. Attributhüllen der Determinanten bestimmen und welche Attr. fehlen
   daraus ⇒ Liste Superschlüssel bestimmen
3. Minimieren Superschlüssel

Schlüsselmenge
KEYS(R) = Menge Schlüsselkandidaten

alternative Schlüssel nicht PK Schlüsselkandidaten

Primattribut $\exists K : Keys(R) : A \in K$ - Teil eines Schlüssels

NPA - Nichtprimattribut -
NPA(R) - Menge dieser

BRUNNEN

Normalisierung Prozess schlechter Rel. Model → gutes durch Zerlegung

Anforderung
1. redundanzfreie Relationen
2. große Relation verlustfrei ableitbar

Kriterien korrekte Zerlegung
1. Verlustlosigkeit
   $r = r_1 \bowtie r_2$
2. Abhängigkeitsbewahrung

1 NF - Erste Normalform

⇔ ∀ Attributwerte atomar
   └ nicht mehrwertig (z.B. Listen) — von Interpretation abhängig

⇒ testbar

VL 2 NF

⇔ in 1NF ∧ $\forall A \in NPA(R): \forall K \in Keys(R): A$ voll abhängig von K

⇒ intuitive Zerlegung

VL 3 NF

⇔ in 2NF ∧ $\forall A \in NPA(R): \forall K \in Keys(R): \nexists Y \in Schema(R): K \to Y \to A$ (transitive FD)

⇒ intuitive Zerlegung

Synthesealgorithmus R mit FD's ↦ äquiv. 3 NF mit Anf.

1. Bestimme „kanonische Überdeckung" aller FDs in R
   Linksreduktion
   Rechtsreduktion
   Entfernung FDs der Form $\alpha \to \emptyset$
   Zusammenfassen gleicher linker Seiten

2. $\forall: \alpha \to \beta$: Erstelle $R := \alpha \cup \beta$

3. Schlüssel

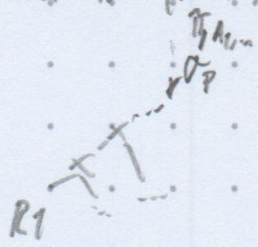
4. Entferne R wenn $\exists R': R \subseteq R'$ ($R' \neq R$)

Anfrageverarbeitung

Anfrage — select $A_{1,..}$ from $R_{1,..}$ where P

↓ A-Compiler

Nicht-optimierter Logischer Plan



↓ A-Optimierer

Optimierter Physischer Plan

↓ Ausführung

Optimierung

Regel-basiert Datenunabhängig
auf algebraischer Ebene
e.g.

1. Selektions-Aufspaltung
   Konjunktion $\sigma_{c_{1\wedge..}}(R) = \sigma_{c_1}(\sigma_{c_2}(..(R)..))$

2. Selections-Pushdown
   kommutativ $\sigma_{c_1}(\sigma_{c_2}(R)) \equiv \sigma_{c_2}(\sigma_{c_1}(R))$
   $\pi_{A_{1,..}}(\sigma_c(R)) \equiv \sigma_c(\pi_{A_{1,..}}(R))$ falls c sich nur auf $A_{1,..}$ bezieht
   $\sigma_c(R \bowtie S) \equiv \sigma_c(R) \bowtie$

3. Join Ersetzung
   $\sigma_c(R \times S) \equiv R \bowtie_c S$

4. Projektions-Pushdown
   $\pi_{L_1}(R) \equiv \pi_{L_1}(\pi_{L_2}(..(R)..))$ falls $L_1 \subseteq L_2 \subseteq ...$
   $\pi(\sigma(R)) \equiv \sigma(\pi(R))$ falls σ noch möglich
   $\pi_L(R \bowtie_c S) \equiv \pi_A(R) \bowtie \pi_B(S)$ falls $A = L \cap R$ ...
   $\pi(R \cup S) \equiv \pi(R) \cup \pi(S)$ π distributiv mit $\cup$

5. Zusammenfassen Operationsfolgen gleicher Op.

weitere Äquivalenzen
$\times, \cup, \cap, \bowtie$ kommutativ
$A, \times, \cup, \cap, \bowtie$ einzeln assoziativ
σ distributiv mit $\cup, \cap, -$

Kostenbasierte Optimierung Datenabhängig
Schätzung durch Kostenmodell

1. Auswahl Join-Reihenfolge
   Schätzung Zwischenergebnis-Größen
   Bottom Up | Tabellen-Größen aus DB-Katalog
   | über Selektivität der Operatoren $\geq 0 \leq 1$
   Stichprobenverfahren
   Selektivität für Stichprobe berechnen
   Histogramm für Attr. Werte in Wertebereich
   evtl. Ausreißer extra; manuelle Änderungen
   multidimensionale Hist. für zusammengesetzte e Bed.

   Schätzung via Annahme Gleichverteilte Attr. Werte

$sel(\sigma_p(R)) = \lvert\sigma_p(R)\rvert / \lvert R\rvert$		$sel(\sigma_{A=konstante}(R)) = 1/\lvert A\rvert$
$sel(R \bowtie S) = \lvert R \bowtie S\rvert / \lvert R \times S\rvert$		$sel(\sigma_{A=B}(R)) = 1/\max(\lvert A\rvert, \lvert B\rvert)$
$p_1 \wedge p_2$ und	$= sel(\sigma_{p_1}(R)) * sel(\sigma_{p_2}(R))$	
$p_1 \vee p_2$ or	$= p_1 + p_2 - (p_1 * p_2)$	
		$sel(R \bowtie_{Fremdschlüssel} S) = \lvert S\rvert / \lvert R \times S\rvert$
		$sel(R \bowtie_{A=B} S) = 1/\max(\lvert A\rvert, \lvert B\rvert)$

2. Auswahl phy. Operatoren

Suchraum
#Binärbaum mit n+1 Blättern = Catalan-Zahl $C_n = (2n)!/((n+1)! * n!)$
Unterschiedliche

⇒ klassischer Ansatz: Dynamische Programmierung
Bottom Up n Phasen je Optimum für Pläne mit k Relationen
Pruning schlechter

Informations Management

Data: organise, structure, query
store, create, modify, delete
interpete, link, analyse, verify

Data Groth of unstructured Data
Science
Business
Industry
World Wide Web

containing implicit information
e.g. for Recomender System, best Products for supermarket
predict subset of items $R \subseteq I$ interessting for user

Machine Learning - ML
task, experience, (feedback)
⇒ perform well

Document Classification
document d, topics $\in T$
⇒ $t \in T$ describing d

NLP - Natural Language Processing | Human Language Technology (focus tech) / Computational Linguistics (analyses)

NLU - understanding transform language → machine representation
G - generation ←

Machine Translation - MT
text in Lang. A
⇒ text in B with same meaning

! Ambiguity
Homograph ↔ Homophone
same writing, diff meaning — voice/Aussprache

Information Retrieval - IR
find subset of document collection D
maximally relevant to query

Extraction - IE
D ↦ structured knowledge

Adverb ↔ Adjective

Pragmatics
utterance

Text Generators
prompt ⇒ continueing/fulfilling text

System Evaluation

intrinsic - in isolation, compare to predefined expectation (gold standard)

extrinsic - in use in larger system/experimental setup

Natural Language Text

Script Σ
Symbol/Character $s \in \Sigma$
String $S = s_1 \dots s_n \in \Sigma^*$
Message string following Grammar
Data exchanged messages (not interpreted)

Text (quasi-) written coherent data of a Language
- coherent: connecting ideas ⇒ semantic meaning

Properties
- type/genre article, ...
- register used lang. variety (dialect, formal, ...)
- style how written (funny, polite, ...)
- domain/topic
- time of writing (lang. changes over time)

Language ↔ Script
- Language: {vocabulary, grammar}
- Script: „alphabet"
  - alphabetic Latin, ...
  - Logographic/syllabic Hanzi, ...
  - consonant-based (Abjab) Arabic, ...
  - segment- (Abugida) Devanagari, ...
  - writing direction

Electronic Text Formats

Plain text	mostly unstructured
HTML, XML	semi-structured e.g. paragraphs
.odt, .docx	XML-based formats e.g. ZIPed XML files
.doc, .rtf	binary content
.pdf	mixed text, binary

Storing text in Relational DBs

with fixed or maximum length
- CHAR(n), VARCHAR(n) typically $\leq$ 4000 8000

arbitrary length
- BLOB - Binary Large Object
  - e.g. for files e.g. images
- CLOB - Character Large Object
  - e.g. text

typically
- no sorting/grouping
- limited comparison (=, <, >, BETWEEN, ...)

Encoding Issues

default of
- source files: IDE
- string types: e.g. UTF-16 in JAVA
- files: OS
- table
- database connection

variable length ⇒ n^{th} char search

Sort - lang. specific

Character Encoding

functions
$enc: \Sigma^* \to \{0,1\}^*$
$dec: \{0,1\}^* \to \Sigma^*$

typically ~~Zeichen~~ characterset
$\psi: \Sigma \to \{0,1\}^*$
$enc(c_1 \dots c_n) = \psi(c_1) \dots \psi(c_n)$

Types
Single Byte $\psi: \Sigma \to \{0,1\}^8$
e.g. ASCII, ISO 8859-1, CP-1252
MultiByte $\psi: \Sigma \to \{0,1\}^{8n}$
e.g. UCS-2, UTF-32
Variable Length depending on char and predefined rules
e.g. UTF-8, UTF-16
Escape-code-based switching between charsets
e.g. ISO 2022

Single Byte

ASCII 7 significant bits, führende 0
control codes
Sonderzeichen
Zahlen ab 30_{16}
Buchstaben groß 41_{16}
klein 61_{16}

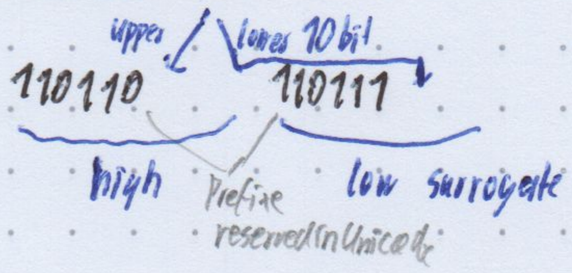
ISO 8859-x Single Byte
lower 128 symbols ⇒ ASCII
upper ⇒ language-specific
e.g. x=1 ⇒ Western Europe
basically one per major lang.

Unicode idea: 1 char set for all inc. logographic, fantasy lang., currency symbols, emojis
4 Byte
21 significant bit
written U+PPxxxx
17 Planes 0x00 to 0x10
PP=0 - BMP - Basic Multilingual Plane
Latin, Arabic, most Chinese, ...
PP=1 - SMP - Supplementary Multilingual Plane
Historic lang., emojis, ...
2 - SIP - Supplementary Ideographic Plane
Extension for Chinese, ...
3-13 empty
14 - SSP - Supplementary Special Purpose
control chars
15-16 - PUA-A/B - Supplementary Private Use Area
Private chars of certain font

UTF-32 4 byte direct Unicode correspondence - slower, bigger than ISO8859-1

UCS-2 2 byte only BMP deprecated

UTF-16 2-4 byte
1 BMP ⇒ direct correspondence
1 supplementary Plane ⇒ subtract 0x10000



UTF-8 1-4 byte direct ASCII correspondence
leading 1s followed by 0 = # bytes (0 statt 10)
continuation bytes start with 10
(theoretically allows upto 6 Bytes)

BRUNNEN

Linguistic Preprocessing

Segmentation

Tokenization input stream (segmenting into) → ordered seq. of tokens
↳ ~ inflected word forms

split at whitespaces?
⚠ cents. $1.25

! e.g. Chinese: no spaces
need context info

Ambiguities

Periods
| sentence termination → seperate token
| abbreviations, ordinal numbers, urls

Comma, Whitespace
| in number

Single quote
| enclosing quote
| contractions, elisions

Dash
| part of token
| ranges

Colon
| sentence delimiter
| time expr.

← **Morphology** - study of word forms/formation

Morphemes - smallest meaning-bearing units

| Free " /Stamm - useable in isolation
| open class words e.g. nouns „cat"
| closed conjunctions

| Bound " / Affix - to words, often for inflection (contracted forms)
| Suffix
| Prefix e.g. -s für Plural
| Infix fan + bloody + tastic e.g. Fugenlaute
| Circumfix get sagt

Morphological Normalization

Stemming - strip word endings
same family → similar stemmed repr.

Errors
Understemming related words → diff. stamm
Over diff words → same stamm

Ambiguity
e.g. Homographs

Lemmatization - undo inflection /to base form
typically requires part of speech
deals with irregular forms

Word Formation/-bildung

Derivation
Stamm + derivation affix

Conversion / zero derivation
change part of speech of stamm without affix

Composition / compounding
Linking Stamms

Decompounding useful for „productive" lang.
e.g. German

Agglutinative lang. combine affixes
e.g. Turkish

BRUNNEN

Analyses of form

Syntax - regularities & constraints word & phrase structure

Part of Speech Tagging (POS Tagging)

↳ ~ lexical class

Tagset ~~of Word classes~~

e.g.

Noun	Preposition	of, by, to
Verb	Pronoun	
Adjective	Determiner	the, a, that
Adverb		

=> valuable info for

- word formation: lemmatization
- possible neighbors: LLM
- pronunciation: speech synthesis
- semantic: word sense disambiguation, shallow parsing

! Ambiguities

Approaces

Rule-based

dictionary: word form → pos tag ~ maybe 90% correct

plus unknown to most common (noun)

iterative (learned) ordered transformation rules

Probabilistic

estimate P(word, context, pos tag)

⇒ most probable

learned from manually labeled training data

Parsing - determine grammatical structure of sentence

Phrase structure grammar

decompose sentence into constituents

↳ word group behaving as single unit

generally older

- Substitution e.g. with it
- Movement as group in sentence
- Coordination group and ...
- Question & yielding constituent

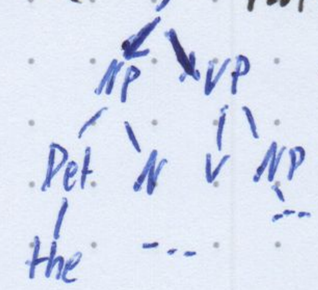
mostly Phrases - word seq. yielding single syntactic Unit

- head ("key" word determines syntactic type
- Grammatical modifier optional, "modify" meaning
  - Pre modifier before head
  - Post " after

e.g.

- NP - Noun Phrase: the black cat
- PP - Prepositional ": in love
- VP - Verb ": eat cheese
- AP - Adjectival ": full of toys
- AdvP - Adverbial ": dearly

Parse tree: S root node



non-terminal (inner) nodes

terminal nodes (leaves)

Bracketed Notation [S [NP ...] ...]

Parenthesized notation (S (NP ...))

Parsing

sentence → in Lang? / generateable by grammar? /

(un-)grammatical sentence?

↳ indicate by * prefix

Syntactic Ambiguity ≥ 1 ~~syn~~ valid sync. structure

Attachement " - constituent can be added to tree at diff places

Coordination " - varying conjunction scope

Garden Path sentence => obvious parse doesn't work

e.g. The old man the boat

↳ Verb

e.g. CFG - context free Grammar

G = (T, N, S, R)

terminals, non-t., startsymbol, production Rules N → (T ∪ N)*

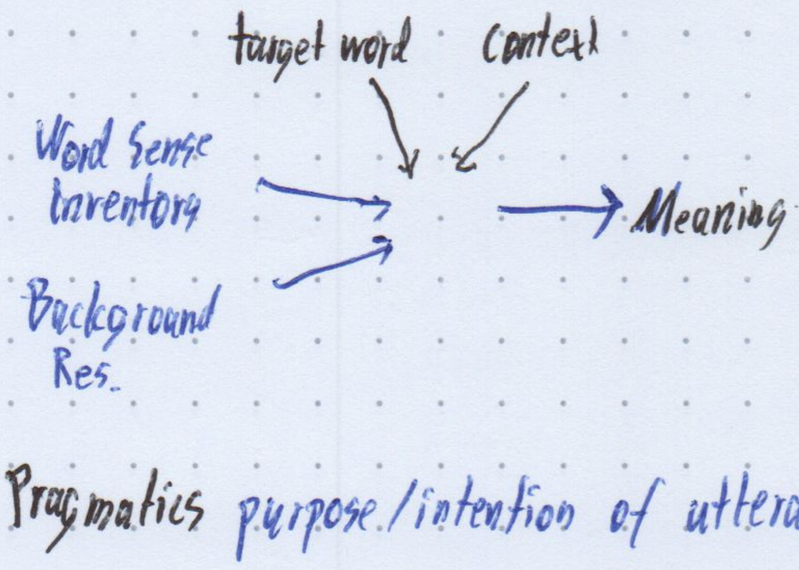
Semantics - meaning { Entities & Rel. / Events

Lexical " - " of lexical items (i.e. words)

Structural " - relationships meaning of lexical items in context

Lexical Ambiguity
syntactic

WSD - Word Sense Disambiguation



Pragmatics purpose/intention of utterance

Text Corpus - Text collection
real word
+ frequency analyses
+ basis for experimental res.

e.g. Brown Corpus - 1961 500 texts POS-tagged
Common Crawl - including copyrighted web
Treebanks - POS, parse tree

Parameters

Language
- Monolingual
- Multi-"
- Parallel - translated texts
  ideally sentence aligned

Communication
- Written
- Spoken
- Mix

Genre / Text type
Domain / Topic
Time of compilation
Size
Static ↔ Dynamic

Annotation on Level - expensive
- Word - POS, lemma, sense - time consuming
- Phrase entities, multi-word exp.
- Sentence syntactic tree, boundaries
- Discourse - co-referential chain
  discourse segments

Workflow
1) retrieve, store original documents
2) convert to plaintext
3) segment
4) annotations
5) format
6) analyse / use

Storing Annotations

Inline - changes original
e.g. likes / POS-tag

Stand-off - seperate file
e.g. at position ... starts ... of length ...

IOB-Annotation Scheme for spans
- I - Begin
- O - Outsite
- Inline - continue

e.g. B O B B I

Corpora in Relational Databases
Words (Word-ID, Word, Frequency)
Word-Index (Word-ID, Sent-ID, Position)
Source-Index (Source-ID, Sent-ID)
Sentences (Sent-ID, Sentence)
Sources (Source-ID, Source)
Bigrams (Word1, Word2, Frequency, Significance)
Co-occurence within sentence (Word1, Word2, Frequency, Significance

Information Retrieval (IR) on unstructured

Information Need / intend
state requiring info to solve problem

⇒ retrieve relevant documents
RE - Retrieval Engine

↔ DBMS on structured data

Query
Repr. info need systeminterpretable

Relevance of document to info need
minimal: about topic
useful / interesting / satisfying user depending
⇒ no clear cut-off

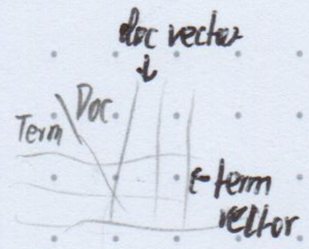
Core Challenges
#results e.g. Millions
Relevance ?
Intend ? suboptimal query ?
Lexical gap ! synonyme phrasen
Ambiguity ! Homographe

Activities
Indexing - document repr. enhancing search
Searching - interpret query, Relevance score, ranking, improve of user feedback

Boolean Retrieval - document ~ set of words
- user needs query syntax
- „feast or famine" - often too few or many results

Query word - document contains word ∈ term
∧, ∨, ¬

Index word ↦ vektor of documents in ~~Matrix~~ Term-Document Matrix { 1 in d, 0 sonst



Search
bitweise ∧ ops res 1 d relevant, 0 not

⚠ large, many 0s
⇒ Inverted Index
word ↦ List of documents
Dictionary Postings
⇒ more optimizations

Vector Space model (VSM) - document ~ Bow - Bag of Words
↳ Multimenge

Ranked Retrieval solves feast or famine
user decides #results
/all ≥ threshold

TF - Term Frequency - #occurences in document $tf_d(t)$
(!) DF - (Inverse) Document Frequency - # docs containing word

query vector in doc vector space

euclidean distance ? - depends on vec. length
⇒ cosine similarity $\frac{q \cdot v}{|q| \cdot |v|}$

Vector weights

$$w_t = \begin{cases} 0 & \text{if } t \notin d \\ 1 & \text{Binary} \\ tf_d(t) & \text{TF} \\ \log(tf_d(t)) + 1 & \text{Normalized TF} \\ \log(tf_d(t)) + 1 \cdot idf(t) & \text{TF} \cdot \text{IDF} \end{cases}$$

eg $2^{\sim 100}$
wirklich linear hilfreicher ?
from 0 to idf
still considers useless determiners (a, the)

IR Evaluation

Error Types in Confusion Matrix

	Relevant	¬ "	ε
retrieved	True Positive	False Pos	
¬ "	False Negative	True Neg	
ε			

P = Precision = $\frac{\text{retrieved} \cap \text{relevant}}{\text{retrieved}} = \frac{\#TP}{\#TP + \#FP}$

R = Recall = $\frac{\text{retrieved} \cap \text{relevant}}{\text{relevant}} = \frac{\#TP}{\#TP + \#FN}$

⚠ achieving one is easy
⇒ F_α score / measure

$F_\alpha = \frac{2PR}{P+R}$ $F_\beta = \frac{(1+\beta^2)PR}{(\beta^2 P) + R}$

harmonic mean

BRUNNEN

Precision at Rank P@R = $\frac{\#\text{relevant docs before } R}{R}$

IE - Information Extraction ↔ IR

Given: D document collection

⇒ structured knowledge
/set of structured facts (entities, events relationships, facts, ...)

often dependent
predefined e.g. SQL
+ for supported query types directly answer for user

easier
Domain-independent
Query types usually unconstraint
faster
- less effective for users goal (the answer)

Entity Recognition

Challenges

Entity ⇔ NoEntity
mobile phone ⇔ Mobile

Coverage Issue
complete list of possible Entitys?

Variation e.g. titels, abbreviations
Mr., Mr. Jr., Dr. First/Lastname A. ...

Ambiguity
Darmstadt (German ⇔ US city)

time dependency
Deutscher Bundeskanzler?

Multi-Word expressions boundaries?
"... Siersch-Stiftung an der TU Darmstadt"

Metonymy figure of speech, Entity by associated Thing
Deutschland für ≈ Fußball Mannschaft

List Lookup Approach

gazetteers - lists of locations

+ simple, fast
+ domain adaptable
- (collect/) Maintain lists
- No Variations Ambiguity

Rule-based

Hand crafted set of often regular expressions

+ fairly good performance
- Labour intensive

mid 1990s ⇒ Supervised Machine Learning from (hand-labeld) database

Corpus-based probabilistic models

probability language structures from large corpora

After IE ⇒ store in Knowledge Base - database for knowledge management
collect, organize, share, info
search, utilize

Knowledge Graphs
fact ~ binary Relationship
e.g. SPO-Triple (subject → predicate → object)
entitys ~ nodes

Science Fiction
/Genre
Star Trek

Automatic Classification

data points = instances (with annotated class labels)
| features ~ vector space coordinates

⇒ learn, generalise
instances → class labels

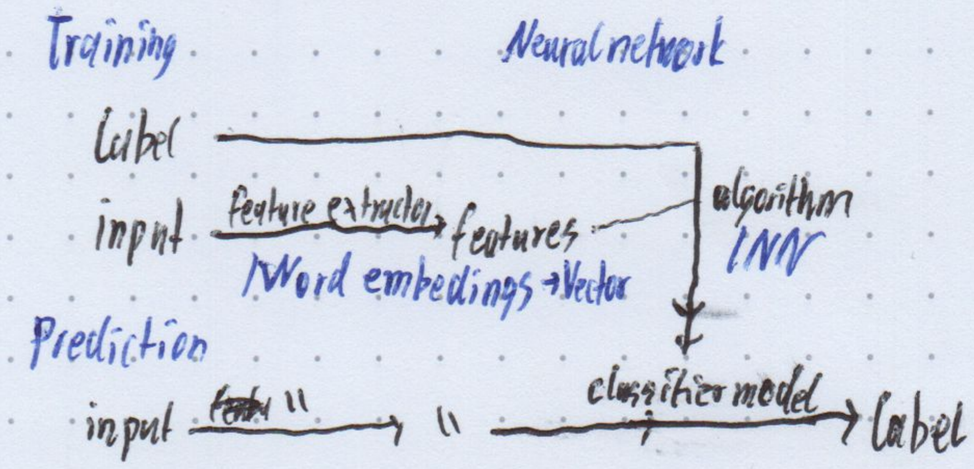
Split the data

Dev-Set
Training-Set
Dev-Test Set - analyse errors, optimize param
Test-Set - no optimization with it of model

binary ↔ multi-class
↳ decomposable in binary
single- ↔ multi-label per instance

sequence classification
instance sequence jointly classified

supervised - trained on labeld instances



BRUNNEN