LZxx Kompressionsverfahren

und

Anwendungsmöglichkeiten

Vorschau

1. Kompressionsverfahren

- **Entwickelt von Abraham Lempel und Jacob Ziv, 1977+, Technion, Haifa, Israel**
- **LZxx ist die Familie der am meisten eingesetzten Kompressionsverfahren, Implementationen sind z.B. *gzip, compress, zip, Stacker, WinZip***
- ***nicht* zu verwechseln mit neueren Verfahren wie z.B. *bzip2***
- **viele verschiedene Unterarten:**

LZ77

- ***Begriffe***

Begriff	*Erklärung*
Input Stream	**„Eingabestrom“: zu komprimierende Sequenz von Zeichen (*character*)**
Character	**„Zeichen“: das zugrundeliegende Datenelement im Eingabestrom (*input stream*)**
Coding Position	**Position des *character's* im Eingabestrom, das gerade kodiert wird (Anfang des *lookahead buffer*'s)**
Lookahead Buffer	**die Zeichenfolge von der *coding position* bis zum Ende des *input stream's***
Window	**Das „Fenster“ der Größe W enthält die W *character* vor der *coding position*, also die letzten W bearbeiteten *character***
Pointer	**verweist auf die letzte Übereinstimmung im Fenster und gibt die Länge derselben an**

- ***Idee***

  - ➔ **Suche nach der längsten Übereinstimmung im *Fenster*, beginnend beim *lookahead buffer***
  - ➔ **gibt einen Zeiger auf die Übereinstimmung zusammen mit dem ersten darauffolgenden *character* des *lookahead buffer's* zurück**
  - ➔ **gibt es keine Übereinstimmung, wird ein „null“-Zeiger und der *character* an der aktuellen Position zurückgegeben**

- ***Kompressionsalgorithmus***

1. setze der *coding position* auf den Anfang des Eingabestroms

2. finde die längste Übereinstimmung im Fenster für den *lookahead buffer*

3. gib das Paar (P,C) mit der folgenden Bedeutung zurück:

- **P ist der Zeiger auf die Übereinstimmung im Fenster**
- **C ist der erste *character* im *lookahead buffer*, für den keine Übereinstimmung gefunden werden konnte**

4. ist der *lookahead buffer* nicht leer, setze das Fenster auf C und die *coding position* auf den *character* nach C und wiederhole ab Schritt 2

- ***Beispiel***

Eingabestrom:

Position	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
character	**A**	**A**	**B**	**C**	**B**	**B**	**A**	**B**	**C**	**X**

Kodierungsablauf:

Schritt	*Position*	*Übereinst.*	*Zeichen*	*Ausgabe*
1	**1**	**--**	**A**	**(0,0) A**
2	**2**	**A**	**B**	**(1,1) B**
3	**4**	**--**	**C**	**(0,0) C**
4	**5**	**B**	**B**	**(2,1) B**
5	**7**	**A B C**	**X**	**(5,3) X**

- ***Dekompressionsalgorithmus***

  ➔ **das Fenster wird gleich behandelt wie bei der Kompression**

  ➔ **bei jedem Schritt wird (P,C) gelesen**

  ➔ **Ausgabe der Sequenz aus dem Fenster, die durch P und C spezifiziert wird**

- ***Eigenschaften***

  - ***Vorteile***

    ➔ **sehr gute Kompressionsrate für viele Arten von Daten**

    ➔ **Dekompression sehr schnell**

    ➔ **geringer Speicherbedarf (gewöhnlich 4-64 Kilobytes)**

  - ***Nachteile***

    ➔ **Kompression kann aufgrund der Suche nach Übereinstimmungen sehr lange dauern**

- ***Implementationen***

  ➔ ***WinZip***

  ➔ ***gzip***

  ➔ ***zip***

  ➔ ***Stacker***

LZ78

- ***Begriffe***

Begriff	*Erklärung*
Charstream	**„Zeichenstrom“: zu komprimierende Sequenz von Zeichen (*character*)**
Character	**Grundlegendes Datenelement in einem *charstream***
Prefix	**„Präfix“: eine Sequenz von Zeichen, die einer anderen vorangeht**
String	**Das *prefix* zusammen mit dem *character*, dem es vorangeht**
Code word	**Grundlegendes Datenelement im *code stream*; es repräsentiert einen *string* aus dem *dictionary***
Code stream	**Die Sequenz der *code words* und *characters* (also die Ausgabe des Kompressionsalgorithmus)**
Dictionary	**Eine Tabelle mit *strings* als Einträge. Jedem *string* wird entsprechend seiner ID im *dictionary* ein *code word* zugeordnet**
Current prefix	**Das *prefix*, das gerade vom Kompressionsalgorithmus bearbeitet wird. Symbol: P**
Current character	**Wird vom Kompressionsalgorithmus ermittelt. Im Allgemeinen ist dies der *character*, dem das aktuelle *prefix* vorangeht. Symbol: C**
Current code word	**Das *code word*, das gerade vom Dekompressionsalgorithmus bearbeitet wird. Es wird mit W bezeichnet, und die *strings*, die es repräsentiert, als *string*.W.**

- ***Idee***

  ➔ **am Anfang ist das *dictionary* leer**

  ➔ **ein neuer *character* wird gelesen**

  ➔ **ist er noch nicht im *dictionary*, wird er diesem hinzugefügt**

  ➔ **ansonsten wird er an den längsten, passenden *string* im *dictionary* angehängt und als neuer Eintrag abgespeichert**

  ➔ **immer längere Einträge im *dictionary***

- ***Kompressionsalgorithmus***

1. *Dictionary* und P sind leer

2. C := der nächste *character* im *charstream*

3. Gibt es den *string* P+C bereits im *dictionary*?

a) ja, P := P+C (erweitere P mit C)

b) nein,

i. gib diese zwei Objekte an den *codestream*:

(1)das *code word*, das P zugeordnet ist (wenn P leer ist, gib 0 zurück)

(2)C, in der gleichen Form wie im *charstream*

ii. füge den *string* P+C dem *dictionary* hinzu

iii.P := leer

4. weitere *character* im *charstream*?

a) ja, gehe zu Schritt 2

b) nein,

i. wenn P nicht leer ist, gib das *code word* aus, das P zugeordnet ist

ii. ENDE

- ***Beispiel***

Eingabestrom:

Position	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
character	**A**	**B**	**B**	**C**	**B**	**C**	**A**	**B**	**A**

Kodierungsablauf:

Schritt	***Position***	***dictionary***	***Ausgabe***
1	**1**	**A**	**(0,A)**
2	**2**	**B**	**(0,B)**
3	**3**	**B C**	**(2,C)**
4	**5**	**B C A**	**(3,A)**
5	**8**	**B A**	**(2,A)**

- ***Dekompressionsalgorithmus***

  ➔ **Zu Beginn ist das *dictionary* leer**

  ➔ **funktioniert genauso wie die Kompression, jedoch wird nicht C, sondern (W,C) gelesen und der entsprechende *string* zurückgegeben**

- ***Eigenschaften***
  - ***Vorteile***

    ➔ ähnlich gute Kompressionsrate für viele Arten von Daten

    ➔ weniger Vergleiche müssen durchgeführt werden als bei LZ77
  - ***Nachteile***

    ➔ Speicherbedarf kann stark wachsen (lange Einträge im Wörterbuch)

- ***Implementationen***

  ➔ ***Compress***

Weitere LZxx-Kompressionsverfahren

- **Lempel-Ziv-Storer-Szymanski (*LZSS*):**
  - **Verbesserung des LZ77-Verfahrens**
  - **setzt bei der Kodierung ein spezielles Bit zur Identifizierung des *code words* setzt und spart somit weiteren Platz**

- **Lempel-Ziv-Welch (*LZW*):**
  - **wichtigste Variante des LZ78-Algorithmus**
  - **Hauptunterschiede zu LZ78:**
    - **das *dictionary* ist niemals leer, sondern wird mit allen *characters* vorinitialisiert**
    - ***one-character-prefix***
    - **der *character*, mit dem ein neues *prefix* beginnt, ist der letzte *character* des vorherigen *strings***

- **Lossy LZW Algorithmus (*LLZW*):**
  - **an GIF-LZW-Algorithmus angelehnt**
  - **Dekompression ist kompatibel**

- **Lempel-Ziv-Miller-Wegman (*LZMW*):**
  - **Variante des LZW-Verfahrens**
  - **hängt anstatt einzelnen Buchstaben den längstmöglichen *string* aus dem *dictionary* an**

2. Pressestimmen

- **berichteten über die Arbeit von Dariao Benedetto, Emanuelle Caglioti und Vittorio Loreto von der *Universita degli Studi di Roma „La Sapienza“*, veröffentlicht in der Fachzeitschrift „Physical Review Letters“**

- **besonders hervorgehoben wurde die Eignung der Methode, um**
  - **die Sprache eines Textes zu erfassen**
  - **den Autor eines Textes zu identifizieren**

- **teilweise wurde gezeigt, daß die Methode nicht für alle Zwecke verwendbar ist (mögliches Interpretationsproblem)**

3. Entropie

Im Folgenden soll die Definition von Chaitin-Kolmogorov gelten:

> **Die Entropie einer Zeichenfolge ist die Länge (in Bits) des kleinsten Programms, welches die Zeichenfolge produziert.**

4. Anwendungen

- **Bei der Analyse einer Zeichenfolge interessiert der Informationsgehalt, z.B.:**
  - ➔ **Teilsequenz eines DNA-Strangs, die ein Gen bestimmt**
  - ➔ **Interpretation und Verstehen eines geschriebenen Texts (Sprache, Thema, Autor, etc.)**
- **Informationsgehalt ist die Entropie**
  - ➔ **Frage: Kann daraus die semantische Information ermittelt werden?**

Vorschau

- **Methode, die Distanz bzw. Ähnlichkeit zweier Zeichenfolgen zu messen**
- **Klärung der Übertragbarkeit auf die Semantik**
- **zumindest bei den durchgeführten Experimenten gelangen**
  - ➔ **Erkennung der Sprache**
  - ➔ **Ermittelung des Autors**
  - ➔ **Erfassung des Themas zum Zwecke der automatischen Klassifikation/ Verschlagwortung**

Distanz

- Durchschnittliche Distanz zwischen zwei aufeinanderfolgende Sequenzen der Form $\sigma_1...\sigma_n$ ist die Ordnung des Inversen ihrer Auftrittswahrscheinlichkeit

- häufiger auftretende Sequenzen brauchen weniger Bytes, seltenere benötigen mehr

- geht nicht deutlich hervor, was gemeint ist:

  a) wird zusätzliche Kodierung (z.B. *Huffman*) bereits einbezogen

  b) je länger die Sequenz, desto größer die Platzersparnis

- genutzte Eigenschaft:

> **Der Quotient aus der Länge eines komprimierten Textes und der unkomprimierten Länge strebt gegen die Entropie des Zeichenstroms.**

Symbol	*Bedeutung*
S	unkomprimierter Text
Z	komprimierter (gezippter) Text
s	Entropie pro Zeichen des Quelltextes

$$\lim\left(\frac{|Z|}{|S|}\right) = s \quad \text{für} \quad |S| \to \infty$$

In anderen Worten:

Der Text wird nicht auf die beste Weise kodiert, aber je länger er ist, umso mehr nähert er sich dem Optimum an.

Relative Entropie

Die relative Entropie gibt an, wieviel Speicherplatz verschwendet wird, wenn eine Zeichenfolge mit einer Methode komprimiert wird, die für eine andere Folge optimiert wurde.

- **Maß für die Länge:**

Symbol	*Bedeutung*
α	ein Zeichenstrom α
β	ein Zeichenstrom β
A	lange Sequenz von α
a	kurze Sequenz von α
B	lange Sequenz von β
b	kurze Sequenz von β
$\|b\|$	Zahl der Zeichen von b
A+b	Konkatenation von b an A, anschließendes komprimieren (zippen)
L_X	Länge eines gezippten Textes X in Bits

Das Maß für die Länge von b in der für A optimierten Kodierung:

$$\Delta_{Ab} = L_{A+b} - L_A$$

- **relative Entropie je Zeichen zwischen α und β:**

$$S_{\alpha\beta} = \frac{(\Delta_{Ab} - \Delta_{Bb})}{|b|}$$

In anderen Worten:

Sind α und β Texte in verschiedenen Sprachen, so ist Δ_{Ab} ein Maß für die Schwierigkeit, die eine Person mit der Muttersprache α hat, einen Text in der Sprache β zu lesen

Eigenschaften des Verfahrens:

- Praktische Versuche brachten eine hervorragende Übereinstimmung mit den theoretischen Werten zustande

- Verfahren sehr robust in Bezug auf Variationen in der Dateigröße, typisch:
  - A: 32-64 Kilobytes
  - b: 1-15 Kilobytes

Anwendung: Spracherkennung

Ziel	**Automatische Erkennung der Sprache eines Textes**
Gegeben	◆ **Text X** ◆ **eine Sammlung langer Texte in sovielen Sprachen wie möglich**
Ablauf	◆ **an alle Texte A_i in allen Sprachen wird ein Auszug x des Textes X angehängt** ◆ **Messung der Unterschiede mittels** $\Delta_i = L_{A_i + x} - L_{A_i}$ ◆ **der Text mit dem kleinsten** Δ_i **ist in derselben Sprache geschrieben**
Versuch	**10 Texte in je 10 Sprachen:** **Dänisch, Deutsch, Englisch, Finnisch, Französisch, Holländisch, Italienisch, Portugiesisch, Schwedisch, Spanisch** ⇨ **Sammlung von 100 Texten**
Ergebnis	◆ **<u>jeder</u> Text X mit einer Länge ≥ 20 Buchstaben Länge wurde der richtigen Sprache zugeordnet**

Anwendung: Autorenidentifikation

Ziel	**automatische Zuordnung eines Textes zu einem Autor**
Gegeben	◆ **Text X, der zugeordnet werden soll** ◆ **eine möglichst große Sammlung von Texten bekannter Autoren in derselben Sprache wie Text X**
Ablauf	◆ **an alle Texte A_i von allen Autoren wird ein Auszug x des Textes X angehängt** ◆ **Suche nach dem Text A_i, dessen Unterschied** $\Delta_i = L_{A_i+x} - L_{A_i}$ **minimal ist**
Versuch	**Sammlung von insgesamt 90 Texten von 11 italienischen Autoren**

Autor	*Textanzahl*	*Erste Wahl*	*Zweite Wahl*
Alighieri	8	8	8
D'Annunzio	4	4	4
Deledda	15	15	15
Fogazzaro	5	4	5
Guicciardini	6	5	6
Macchiavelli	12	12	12
Manzoni	4	3	4
Pirandello	11	11	11
Salgari	11	10	10
Svevo	5	5	5
Vega	9	7	9
Summe	*90*	*84*	*89*

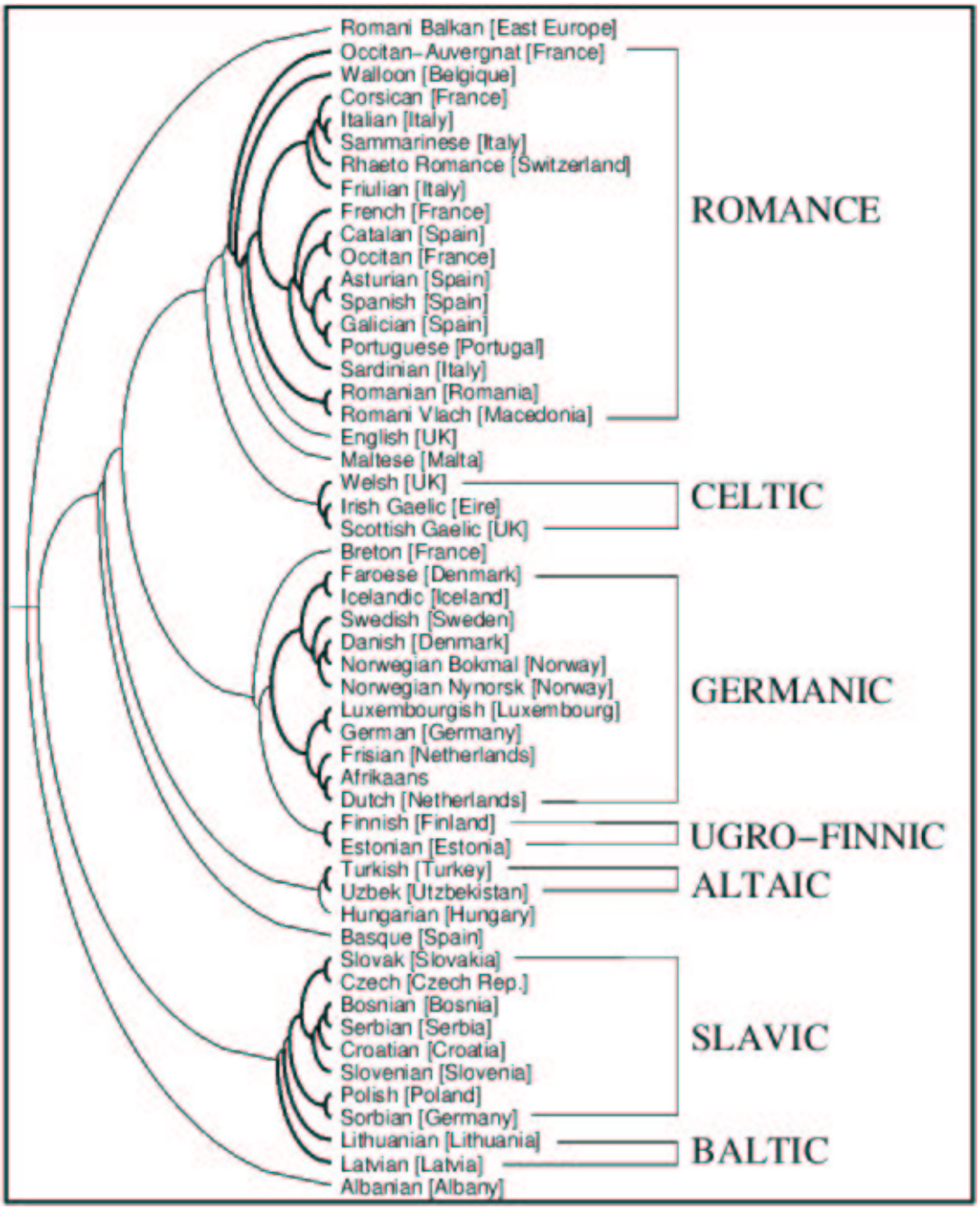
Ergebnis	◆ **Eindeutige Zuordnung des Textes in 93,3% der Fälle („erste Wahl“)** ◆ **in der engen Auswahl in 98,9% der Fälle (als schlechtestenfalls „zweite Wahl“)**

<u>Bemerkung:</u>

- **Erfolgsrate durch z.B. Gewichtung der Durchschnitte der ersten *m* als wahrscheinlich eingestuften Kandidaten des Textes noch steigerbar**
- **Faktoren wie z.B. Variationen im Schreibstil beeinflussen die Erfolgsrate**

Anwendung: Sequenzklassifikation

Ziel	**Klassifikation von Sprachen anhand von Texten**
Gegeben	**eine möglichst große Sammlung von Texten in jeweils sovielen Sprachen wie möglich**
Probleme	◆ **Verfügbarkeit derselben langen Texte in vielen verschiedenen Sprachen** ◆ **einheitliche Kodierung des Textes**
Ablauf	◆ **jeder Text in jeder Sprache wird an jeden der anderen Texte angehängt** ◆ **Konstruktion einer Distanzmatrix, deren Elemente die Distanzen zwischen zwei Texten sind:** $$S_{\alpha\beta} = \frac{(\Delta_{Ab} - \Delta_{Bb})}{\Delta_{Bb}} + \frac{(\Delta_{Ba} - \Delta_{Aa})}{\Delta_{Aa}}$$ ◆ **Normalisierung mit** Δ_{Aa} **bzw.** Δ_{Bb} ◆ **Entropie keine Distanz im mathematischen Sinne** ⇨ **Matrixelemente sollen die Dreiecksungleichung erfüllen** ◆ **Erzeugung des Sprachbaums aus der Distanzmatrix mittels der *Margoliaschen Methode* aus dem Softwarepaket *PhylIP* (*Phylogeny Inference Package*)**
Versuch	◆ **In 52 europäischen Sprachen: „Die Allgemeine Erklärung der Menschenrechte“** ◆ **Kodierung der Texte in UNICODE**



Ergebnis	**Alle linguistischen Hauptgruppen Romanisch, Keltisch, Germanisch, Ugro-Finnisch, Slavisch, Baltisch und Altaisch wurden erkannt**

Zusammenfassung

- **Verschiedene Anwendungen wurden für eine sehr allgemeine Methode, Sequenzen zu erkennen und zu klassifizieren, gezeigt**
- **weitere Anwendungen, z.B. Ermittlung des Informationsgehalt (s. folgendes Beispiel) sind denkbar (Aktienkurse, DNA, etc.)**

Bemerkung

Die Arbeit von Benedetto, Caglioti und Loreto ist nicht die erste, die mit mathematischen Methoden literarische Texte auf Autorenschaft prüft:

- **Linguistikprofessor George Zipf, 1932, Harvard-Universiät: „Studien zur Wortfrequenz“**
- **George Yule, Schottland, 1944: ordnete in seinem Werk „The Statistical Study of Literacy“ „De imitatione Christi“ aus dem 15. Jh. dem Mystiker Thomas a Kempis zu**
- **amerikan. Statistiker R. Frederik Morteller und David L Wallace, 1964: gaben Aufschluß über die Autoren der „Federal Papers“ aus dem 18. Jh.**

Praxisbeispiel: Versuche eines Journalisten

Versuch 1: Informationsgehalt

Gegeben:

- **18 Texte, 14000 Wörter, 105000 Zeichen**
- **häufigstes Thema: „Israel“**
- ***WinZip* als Kompressionsprogramm**

1. reine Kompression:

➔ **Kompression auf $\frac{1}{3}$ der ursprünglichen Größe, also $\frac{2}{3}$ Redundanz**

2. Sortierung:

➔ **Kompression auf $\frac{1}{5}$ der ursprünglichen Größe, also hat sein Text etwa 15% mehr Relevanz als ein Wörterbuch**

3. Randomisierung:

➔ **Kompression auf $\frac{2}{5}$ der Größe, eine zufällige Sammlung hat also mehr Gehalt als die verfassten Artikel**

Versuch 2: Autorenidentifikation und Vergleich

Gegeben:

- **je ein langer (1000 Wörter) und ein kurzer (50 Wörter) Text von zwei Journalisten**
- ***WinZip* als Kompressionsprogramm**

1. Identifizierung des Autors:

➔ korrekte Identifizierung des Autors

2. Vergleich kurze mit langen Artikeln:

➔ außerordentlich gute Kompression, also nur wenig neues in den Artikeln

Bemerkung:

- **Nicht korrekte Anwendung des Verfahrens**
- **oben definierter Entropiebegriff lässt sich nicht ohne weiteres auf die Semantik, bzw. den Informationsgehalt, übertragen**

5. Quellen

- **„Language Trees and Zipping“, Benedetto, Caglioti, Loreto,**

  „La Sapienza“ Universität, Rom, Italien, 13.05.2002

- **„Buchstabenstauchung“, Neue Züricher Zeitung, 31.05.2002,**

  http://www.nzz.ch

- **Datenkompressionen,**

  http://www.datacompression.info/Lossless.shtml,

  http://www.rasip.fer.hr/research/compress/algorithms/fund

- **„Zip-Algorithmus identifiziert Autoren“, Heise Newsticker, 28.01.2002,**

  http://www.heise.de/newsticker