Ambiguität, Wahrscheinlichkeit, Kodierung

Christian Wurm
Universität Düsseldorf

1. Einleitung
   - Der linguiststische Rahmen

2. Kodierungstheorie
   - Allgemeine Grundlagen und Beispiele
   - Ein realistischeres Szenario
   - Das Kompressionsschema
   - Die geteilte Information reduzieren

Ambiguität und Kodierung

Wir haben letzte Woche besprochen, wie Ambiguität die Kommunikation effizienter machen kann:

eine Nachricht hat mehrere Bedeutungen

$\Longrightarrow$ ich brauche weniger Nachrichten (für dieselbe Anzahl Bedeutungen)

$\Longrightarrow$ ich kann meine Nachrichten (als Atome) kürzer *kodieren*

$\Longrightarrow$ ich muss meinen Mund weniger bewegen um sie zu sagen

Mich persönlich überzeugt das voll und ganz – nur dass wir den eigentlichen Punkt – die Kodierung – bislang außen vor gelassen haben.

Das holen wir heute nach.

Also: wir betrachten heute Kodierungen, konkret.
Die erwartete Länge einer Kodierung ist definiert als

$$\text{Länge eines Kodewortes} \cdot \text{Wahrscheinlichkeit eines Kodewortes} \tag{1}$$

Üblicherweise kodiert man in $\{0, 1\}^*$, und wir machen das auch so.

Kodierungstheorie hängt eng mit Wahrscheinlichkeitstheorie zusammen, wie wir sehen werden. Die Frage von Juba et al. ist:

Frage

Wie sieht eine **optimale Kodierung** aus unter Rahmenbedingungen, die in etwa denen natürlicher Sprache entsprechen?

Ein Beispiel

(1) Peter sagte Klaus dass Gabi ihn angelogen hat.

Dieser Satz ist natürlich ambig; man kann ihn auf zwei Arten disambiguieren:

(2) a. Peter sagte Klaus dass Gabi ihn, also Klaus, angelogen hat.
b. Peter sagte Klaus dass Gabi ihn, also Peter, angelogen hat.

Das ist natürlich länger und mühsamer.

Die Frage ist: lohnt die Mühe? Die Antwort ist: Das hängt davon ab, was wir an Hintergrund voraussetzen können!

Ein Beispiel

Die Antwort ist: Das hängt davon ab, was wir an Hintergrund voraussetzen können!

Z.B.: Wir wissen: Klaus hat nie mit Gabi geredet. Dann ist die Sache klar.

Z.B.: Wir wissen nichts. Dann ist die Sache unklar.

Aber: wer ist "wir"?

Ein Beispiel

A Sprecher und Hörer. In diesem Fall ist es einfach die optimale Kodierung zu entscheiden.

B Nur Sprecher/nur Hörer: in diesem Fall ist die Sache problematisch! Dieser Fall lässt sich genauer umschreiben mit: Hörer und Sprecher haben verschiedenes Hintergrundwissen, und keine genaue gegenseitige Kenntnis davon!

Fall B ist eigentlich die linguistisch relevante Fall. Es ist gleichzeitig der Fall, der in der Kodierungstheorie normalerweise nie berücksichtigt wird!

Grundlagen der Kodierung

Seien Σ, T zwei Alphabete (z.B. $T = \{0, 1\}$). Ein Kode (von Σ in T) ist Paar

$$(\phi, X),$$

wobei

$$X \subseteq T^*, \text{ und } \phi : \Sigma \to X$$

eine Bijektion ist, so dass die homomorphe Erweiterung von

$$\phi : \Sigma^* \to X^*$$

weiterhin eine Bijektion ist.

Beispiele

Bsp. 1:
$\Sigma = \{a, b, c, d\}$, $X = \{0, 00, 1, 11\}$

$\phi(a) = 0$; $\phi(b) = 1$; $\phi(c) = 00$; $\phi(d) = 11$

$\phi(aa) = \phi(a)\phi(a) = 00 = \phi(c)$ – Keine Bijektion, also keine Kodierung!

Bsp. 2
Σ s.o., $X = \{00, 01, 10, 11\}$

Kodierung? Ja, ein sog. Block-Code.

Zwei Arten von Codes

Man unterscheidet 2 Arten von Codes:

1. Sog. universelle Codes: damit kann man jedes Dokument kodieren, ohne eine Verteilung zu kennen
2. Probabilistische Codes: diese basieren auf einer zugrundeliegenden Wahrscheinlichkeitsverteilung

Natürlich sind universelle Codes universeller: man kann sie jederzeit und Überall einsetzen (Lempel-Ziv, aka. .zip)

Aber ein universeller Code wird niemals an die Kompression eines probabilistischen Codes heranreichen – vorausgesetzt eben, es *gibt* diese zugrundeliegende Wahrscheinlichkeitsverteilung!

Zwei Arten von Codes

Wir betrachten nun erstmal einen klassischen probabilistischen Code, den sog. Huffmann-Code.

Dieser Code ist sehr bekannt, weil er eng verknüpft ist mit der Entropie einer Verteilung!

Präfixfreie Codes

Ein Kode ist **präfixfrei**, falls es kein $x, y \in X$ gibt so dass

$$xz = y, \text{ für beliebiges } z \in T^+ \ (T = \{0, 1\}).$$

Z.B.: $\phi(a) = 01, \phi(b) = 11, \phi(c) = 011, \phi(d) = 110$ ist eine Kodierung, aber nicht präfixfrei. Dekodiere:

01111111 - abbb
01111110 - cbd

Optimale Kodierungen

Bei jeder Kodierung möchten wir, dass jede Kodierung eines Textes möglichst kurz wird.
Nehmen wir an, wir haben eine Wahrscheinlichkeitsverteilung

$$P : \Sigma \to [0, 1] \tag{2}$$

über Σ, und weiterhin, dass die Wahrscheinlichkeiten der Worte unabhänging voneinander sind. Das bedeutet:

- wenn ein Buchstabe sehr wahrscheinlich ist, dann wollen wir ihn kürzer kodieren,
- wenn er unwahrscheinlich ist, dann länger.

Optimale Kodierungen

Sei $w \in \Sigma^*$. Wir bauen uns eine Zufallsvariable L, so dass

$$L(a) = |\phi(a)| \tag{3}$$

Ziel: wir möchten den Erwartungswert von X

$$\mathrm{E}(L) = \sum_{a \in \Sigma^*} |\phi(a)| \cdot P(a) \tag{4}$$

möglichst minimieren! (Man nennt das inversen Kompressionsfaktor)

Optimale Kodierungen

Jeder Buchstabe im Ausgangsalphabet Σ hat Länge 1; er wird – nach Erwartungswert – in der Kodierung im Schnitt mit $\mathrm{E}(L)$ Symbolen ersetzt. Deswegen nennen wir die *Inversion*

$$\frac{1}{\mathrm{E}(L)} \text{ den } \textbf{Kompressionsfaktor}$$

der Kodierung. Ein wichtiger Punkt ist nun:

$\mathrm{E}(L)$ *kann niemals kleiner sein als die Entropie* $H(P)$.

Das bedeutet wir müssen jedes Symbol im Schnitt mit mindestens $H(P)$ Zeichen kodieren.

Optimale Kodierungen

Es gibt einen einfachen Algorithmus, den sogenannten **Huffman code**, der folgendes liefert:

- Eingabe: ein beliebiges Alphabet Σ mit einer zugehörigen Wahrscheinlichkeitsfunktion $P : \Sigma \to [0, 1]$
- Ausgabe: eine Kodierung von Σ in $\{0, 1\}^*$ in einem Präfix-freien Kode mit maximalen Kompressionsfaktor (es gibt aber immer mehrere solcher Kodierungen).

Wir zeigen den Algorithmus, der eigentlich sehr einfach ist, mit einem Beispiel.

Der Huffmann Code – Beispiel

Nehmen wir an, $\Sigma = \{a, b, c, d\}$, mit folgenden Wahrscheinlichkeiten (bzw. Häufigkeiten):

- $P(a) = 0.1$
- $P(b) = 0.2$
- $P(c) = 0.3$
- $P(d) = 0.4$

Der Huffmann Code – Beispiel

Wir fangen damit an, das Buchstabenpaar zu nehmen, das am seltensten vorkommt. Das ist natürlich

$$\{a, b\} \text{ mit } P(\{a, b\}) = 0.3.$$

Wir ersetzen nun

$$\{a, b\} \mapsto \{x_1\},$$

so dass unser neues Alphabet ist

$$\{x_1, c, d\}, \text{ wobei } P(x_1) = 0.3.$$

Der Huffmann Code – Beispiel

Nun machen wir ebenso weiter: im neuen Alphabet ist das Buchstabenpaar mit der geringsten Wahrscheinlichkeit $\{x_1, c\}$, also ersetzen wir

$$\{x_1, c\} \mapsto \{x_2\}$$

mit dem resultierenden Alphabet

$$\{x_2, d\}, \text{ wobei } P(x_2) = P(\{x_1, c\}) = 0.6.$$

Nun machen wir den Schritt ein letztes Mal: das resultierende Alphabet ist

$$\{x_3\} \text{ mit } P(x_3) = 1.$$

Der Huffmann Code – Beispiel

Nun "entpacken" wir das ganze wieder.

- Wir nehmen an, x_3 wird kodiert durch das leere Wort ϵ.

ϵ steht dann aber eigentlich für 2 Buchstaben: x_2 und d. Das erste ist wahrscheinlicher.

- Also kodieren wir x_2, indem wir eine 0 an unser Kodewort hängen, d mit einer 1.

Nun steht x_2 (bzw. 0) wiederum für zwei Buchstaben: x_1, c.

- Wir setzen $x_1 = 00, c = 01$ (in diesem Fall ist es egal, die Wahrscheinlichkeiten sind gleich).

Nun dasselbe mit x_1 (bzw. 00);

- in diesem Fall bekommen wir 000 für b, 001 für a.

Der Huffmann Code – Beispiel

Wir bekommen also:

- $\phi(a) = 001$
- $\phi(b) = 000$
- $\phi(c) = 01$
- $\phi(d) = 1$

Wir nehmen nun L wie oben, und bekommen:

$$\mathrm{E}(L_\phi) = 0.1 \cdot 3 + 0.2 \cdot 3 + 0.3 \cdot 2 + 0.4 \cdot 1 = 1.9 \tag{5}$$

Huffmann Code – Block Kode

$$E(X_\phi) = 0.1 \cdot 3 + 0.2 \cdot 3 + 0.3 \cdot 2 + 0.4 \cdot 1 = 1.9 \tag{6}$$

Der Kompressionsfaktor ist also $\frac{1}{1.9}$. Natürlich kommt dasselbe raus, wenn wir im Kode einfach 0 und 1 vertauschen. Wenn wir das vergleichen mit dem folgenden Block-Kode

- $\chi(a) = 00$
- $\chi(b) = 01$
- $\chi(c) = 10$
- $\chi(d) = 11$

(der auch Präfix-frei ist), dann bekommen wir

$$E(X_\chi) = 0.1 \cdot 2 + 0.2 \cdot 2 + 0.3 \cdot 2 + 0.4 \cdot 2 = (0.1 + 0.2 + 0.3 + 0.4)2 = 2 \tag{7}$$

Der Kompressionsfaktor beträgt also nur $\frac{1}{2}$.

Kompression und Entropie

Wie ist die Entropie für P?

$$H(P) = -(0.1log_2(0.1)+0.2log_2(0.2)+0.3log_2(0.3)+0.4log_2(0.4)) = 1.84643 \quad (8)$$

Nehmen wir an, dagegen an dass

$$P'(a) = \ldots = P'(d) = 0.25$$

ändert sich die Lage: in diesem Fall ist natürlich χ die optimale Kodierung. Die Entropie ändert sich wie folgt:

$$H(P') = -(4 \cdot (0.25log_2(0.25)) = 2 \quad (9)$$

Die Entropie ist größer, daher wird auch die Kompressionsrate schlechter sein. Am Ende gilt:

Kompression und Entropie

Sei P eine Wahrscheinlichkeitsverteilung über Σ, ϕ ein (beliebiger) Kode über $\{0, 1\}$. Dann kann der Kompressionsfaktor von ϕ niemals grösser sein als $\frac{1}{H(P)}$, berechnet zur Basis 2. Noch einfacher:

$$H(P) \leq \mathrm{E}(L_\phi) \tag{10}$$

Die Entropie ist also die untere Schranke dafür, wie kompakt wir kodieren können. Für die Huffmann Kodierung X_H gilt übrigens auch:

$$H(P) \leq \mathrm{E}(L_H) \leq H(P) + 1 \tag{11}$$

Die Voraussetzungen

Die Voraussetzungen für eine Huffmann Kodierung ist allerdings eine Verteilung P, die sich beide Seiten **teilen**:

Ausreichende Bedingung

Für Hörer H, Sprecher S gilt: wenn Σ und P bekannt sind, dann können H, S erfolgreich mit dem Huffmann Kode kommunizieren – die Kodierung ist eindeutig! (man muss aber noch einige Parameter setzen)

Genau das ist aber in normaler Kommunikation nicht gegeben! Wir müssen uns also was anderes überlegen...

Kurzer Exkurs: Universeller Kodierung

Eine Kodierung ist universell, wenn Sie keine Wahrscheinlichkeitsverteilung braucht.

Gute universelle Kodierungen sind *adaptiv*, d.h. sie passen sich dem Eingabedokument an.

Ausserdem soll sie *universell dekodierbar* sein, also ohne ein "Wörterbuch" auskommen.

Die berühmteste universelle Kodierung ist Lempel-Ziv.

Der Lempel-Ziv Code

(Meist Lempel-Ziv-Welch) Lempel-Ziv ist die Grundlage von .zip und vielen anderen (verlustfreien) Formaten!
Intuitiv funktioniert der Kode wie folgt:

- Jede Zeichenfolge (Länge 2) bekommt ein Kürzel zugewiesen.
- Bei Wiederholung wird sie durch das Kürzel ersetzt
- Das wird iteriert appliziert, so dass auch Folgen beliebiger Länge Kürzel zugewiesen bekommen (vorausgesetzt, sie bestehen aus zwei Teilfolgen, die bereits ein Kürzel haben).

Hieraus folgt: je repetitiver ein Text, desto kompakter wird er kodiert!

Der Lempel-Ziv Code

Der LZ-Kode ist wichtig für uns, weil er "dynamisch" *während* der Text verarbeitet wird entsteht.

Er gibt also ein gutes Mass nicht für die Gesamtentropie eines Textes (wie der Huffman-Code), sondern für die bedingte Entropie neuer Worte.

Wir möchten nun ein Szenario für Kodierungen betrachten, welches näher an der linguistischen Realität liegt.

Ein realistischeres Szenario

Unsere Kodierungen gingen bislang ein wenig an der Realität vorbei. Hier sieht die Sache etwas anders aus:

- $\phi^{-1}(x)$ ist nicht eindeutig, also ϕ keine Bijektion.
- Es gibt eine *apriori* Wahrscheinlichkeitsverteilung über Σ. Aber: die ändert sich, wenn neue Information kommt.
- Also wird dekodiert nach dem Schema (separate Dekodierung):

$$\psi(x) = \underset{a:\phi(a)=x}{argmax}\ P(a|C) \tag{12}$$

Da zwischen H und S Informationen ausgetauscht werden in der Kommunkation, ändert sich die Wahrscheinlichkeit mit dem Kontext C.

Ein realistischeres Szenario

Die Kodierung läuft parallel dazu:

$$\phi(a) = \underset{x:\psi(x)=a}{argmin} |x| \tag{13}$$

Das ist eindeutig! Übermittelte Botschaften ändern natürlich die Verteilung $P(-|C)$ –

Aber: falls beide vom gleichen *apriori* ausgehen, bleiben die Verteilungen gleich – Kodierung und Dekodierung sichern uns eine eindeutige Zuordnung!

Ein Beispiel: Grundlage

Nehmen wir an, wir haben zwei Bedeutungen: $R = \{r_1, r_2\}$ und dwei komplexe Ausdrücke ab, abc, abd. Es gilt:

$$\begin{aligned} \phi^{-1}(ab) &= \{r_1, r_2\} \\ \phi^{-1}(abc) &= \{r_1\} \\ \phi^{-1}(abd) &= \{r_2\} \end{aligned}$$

Wir nehmen nun an, Kommunikation findet ständig in einem Kontext C statt, und es gilt:

$$P(r_1|C) > P(r_2|C) \tag{14}$$

Dann bekommen wir:

$$\phi(r_1|C) = ab$$
$$\phi(r_2|C) = abd$$

Ein Beispiel – weitergesponnen

Der Sprecher kodiert also eindeutig, der Hörer dekodiert ebenso eindeutig.

Nimm an, es wird eine Nachricht übermittel.

Das Resultat ist also ein *neuer Kontext* C', denn unsere Information ist aktualisiert. Wir haben

$$P(r_1|C') < P(r_2|C') \tag{15}$$

Dann bekommen wir:

$$\phi(r_1|C') = abc$$
$$\phi(r_2|C') = ab$$

Ein weiteres Beispiel: Mathematik

Auch mathematische Notationen sind nicht immer komplett eindeutig, siehe das folgende Beispiel:

$$f(x) = log\ x/x \tag{16}$$

Das ist ambig wegen fehlender Klammern. Wenn der Mathematiker aber schreiben wollte $log(x/x)$, dann würde er schreiben: 0 (Mathematiker sind faul)

Also: $(log\ x)/x$.

Und genau dieser Logik folgen wir hier!

Adaptive Kodierung – die Extremfälle

Wenn wir nun adaptiv kodieren, sieht man, dass Huffmann einerseits und Lempel-Ziv andererseits Extreme sind:

- Die Huffmann Kodierung würde nach jedem Kontext-Update alle Worte neu kodieren: andere Wahrscheinlichkeit, andere (optimale) Kodierung!
- Lempel-Ziv würde alle Worte invariant kodieren: die Wahrscheinlichkeit spielt keine Rolle, also spielt der Kontext keine Rolle!

Adaptive Kodierung – die Extremfälle

Wie gesagt haben beide Extreme Vor- und Nachteile:

- Huffmann ist extrem sensibel bezüglich Wahrscheinlichkeiten: wenn Hörer und Sprecher Wahrscheinlichkeiten subjektiv verschieden einschätzen, haben Sie keine Chance sich zu verstehen
- Lempel-Ziv ist eher ineffizient: Kontextinformation kann nicht genutzt werden!

Unser Beispiel in der Mitte

Unser Beispiel:

$$\begin{aligned} \phi^{-1}(ab) &= \{r_1, r_2\} \\ \phi^{-1}(abc) &= \{r_1\} \\ \phi^{-1}(abd) &= \{r_2\} \end{aligned}$$

ist in der Mitte der beiden Extreme anzusiedeln:

- Kodierung basiert auf einem festen Wörterbuch,
- aber wegen der Uneindeutigkeit kann es verschiedenen Kodierungen derselben Bedeutung kommen, abhängig vom Kontext!

Ich denke, dieses Beispiel kommt der linguistischen Realität am nächsten! Bis auf eine Kleinigkeit...

Adaptive Kodierung: Die Zusammenfassung

Wir haben also eine Kodierung, die sich mit der zugrundeliegenden Wahrscheinlichkeitsverteilung *adaptiert*:

- Äußerungen ändern Kontexte
- Kontexte ändern Kodierungen (also Äußerungen)
- Aber: die Kodierung bleibt jederzeit eindeutig, *weil sie es am Anfang ist*.

Die adaptive Kodierung, und das Problem

Und genau diese letzte Annahme ist unrealistisch in natürlicher Kommunikation:

- Zwei Menschen werden evtl. ähnliche apriori Annahmen über Wahrscheinlichkeiten haben,
- aber wohl niemals dieselben!
- Das Szenario, das wir hier gezeichnet haben, führt als in diesem Fall zu einer kommunikativen Katastrophe!

Das Problem

Wie können wir kodieren, wenn die *apriori* Wahrscheinlichkeiten von Sprecher und Hörer verschieden sind?

Das wollen wir besprechen.

Juba et al.

... besprechen eine Kodierung, die

- adaptiv ist an eine sich ändernde Verteilung
- aber nicht voraussetzt, dass Sprecher und Hörer *dieselbe* Verteilung haben
- stattdessen gibt es einen Parameter α, der bestimmt, in welchem Rahmen sich Hörer und Sprecher unterscheiden, der die Kodierung bestimmt
- darüber hinaus müssen Hörer und Sprecher noch weitere Parameter teilen (das Wörterbuch/die Grammatik)

Aber eines nach dem anderen.

Wahrscheinlichkeitsverteilungen

Wir erinnern uns:

Bayesianische Wahrscheinlichkeit

Eine Wahrscheinlichkeit ist eine subjektive Zuschreibung von Plausibilität. Dementsprechend erwarten wir, dass zwei verschiedene Individuen denselben Eregnissen verschiedene Wahrscheinlichkeiten zuweisen (basierend u.a. auf unterschiedlichen Hintergrundannahmen)

Aber:

Im Normalfall ist der Unterschied in der Zuweisung zwischen kommunizierenden Individuen überschaubar. Hierauf basiert der Ansatz von Juba et al.

α-Nähe

Ich halte mich im Folgenden an die Notation des Artikels.
Sei M ein Menge der Bedeutungen (meaning, message,...)
Wir benennen $\Delta(M)$ die **Menge aller Wahrscheinlichkeitsverteilungen**

$$p : M \to [0, 1]$$

Definition

Zwei Verteilungen $p, q \in \Delta(M)$ sind α-nah (für $\alpha \geq 1$), falls gilt: f.a. $m \in M$,

$$p(m) \leq \alpha q(m), \text{ und } q(m) \leq \alpha p(m).$$

Wir schreiben dann $p \approx_\alpha q$.
Das bedeutet: p, q dürfen sich maximal um Faktor α unterscheiden bei der Zuweisung von Wahrscheinlichkeiten.

Definition

Zwei Verteilungen $p, q \in \Delta(M)$ sind α-nah (für $\alpha \geq 1$), falls gilt: f.a. $m \in M$,

$$p(m) \leq \alpha q(m), \text{ und } q(m) \leq \alpha p(m).$$

Wir schreiben dann $p \approx_\alpha q$.
Das bedeutet: p, q dürfen sich maximal um Faktor α unterscheiden bei der Zuweisung von Wahrscheinlichkeiten.

α ist ein wichtiger Parameter der Kodierung, und insbesondere muss α dem Sprecher bekannt sein – sonst wird seine Nachricht falsch dekodiert!

Probabilistische Kodierungsschema

Ein **probabilistische Kodierung** ϕ für M nimmt als Eingabe:

1. Eine Verteilung $p \in \Delta(M)$
2. ein $m \in M$

und liefert als Ausgabe die Kodierung $\phi(m) \in \{0,1\}^*$.

Zu jedem Kodierungsschema kommt ein Dekodierungsschema ψ (wir nennen es explizit anders hier!)

Probabilistische Kodierungsschema

Eine **probabilistische Dekodierung** ψ für M nimmt als Eingabe:

1. Eine Verteilung $q \in \Delta(M)$
2. ein $x \in \{0,1\}^*$

und liefert als Ausgabe die Dekodierung $\psi(x) \in M$.

Soviel ist klar. Wichtig ist aber: Kodierung und Dekodierung müssen übereinstimmen!

$$\psi(\phi(m,p),p) = m \tag{17}$$

Parameter und Kodierungsschemata

Huffmann und Lempel-Ziv Schema sind *parameterfrei*: gegeben p, M ist die Kodierung eindeutig.
Unser linguistisches Beispiel war anders: wir hatten eine Zuordnung

$$\begin{aligned} \phi^{-1}(ab) &= \{r_1, r_2\} \\ \phi^{-1}(abc) &= \{r_1\} \\ \phi^{-1}(abd) &= \{r_2\} \end{aligned}$$

Das entspricht einer Bedeutungsmatrix wie bei Ferrer& Solé, oder äquivalenten einem **bipartiten Graphen**, einer uneindeutigen Zuordnung

Ausdruck $- - --$ Bedeutung

Juba et al. verfolgen diesen Ansatz.

Kodierungsschema

Ein solches "Wörderbuch" denotieren sie mit θ; und ϕ, ψ kodieren bzw. dekodieren auf Basis von θ; man schreibt dann ϕ_θ, ψ_θ.

Definition

Ein **Kodierungsschema** *ist ein Tupel* (M, Θ, ϕ, ψ)*, wobei gilt: f.a.* $\theta \in \Theta$*,* $p \in \Delta(M)$*:*

$$\psi_\theta(\phi_\theta(m, p), p) = m \tag{18}$$

Soweit so gut. Jetzt wird es interessant:

Definition

Ein Kodierungsschema (M, Θ, ϕ, ψ) *ist* α*-robust, falls gilt: für jedes* α*-nahe Paar* p, q *haben wir*

$$\psi_\theta(\phi_\theta(m, p), q) = m \tag{19}$$

Die interessante Kennzahl ist die (inverse) **Kompression** des Schemas; das ist die inverse erwartete Länge

$$E_{m \sim p}(|\phi_\theta(m, p)|) \tag{20}$$

wobei

- $|-|$ die Länge des Kodewortes $|\phi_\theta(m, p)|$ ist
- $E_{m \sim p}(-)$ bedeutet: M verteilt sich nach p, also:

$$E_{m\sim p}(|\phi_\theta(m,p)|) = \sum_{m\in M}(|\phi_\theta(m,p)| \cdot p(m)) \tag{21}$$

Diesen Term gilt es also zu minimieren.

Juba et al. abstrahieren auch vom konkreten θ, indem sie eine Wahrscheinlichkeitsverteilung μ über $\theta \in \Theta$ definieren.

Diesen Schritt müssen wir erstmal nicht mitgehen.

Kodierung/Dekodierung

Man beachte, dass solche Schemata normalerweise von der Dekodierung ausgehen: nimm an, ψ ist gegeben. Dann ist

$$\phi(m) = \underset{x:\psi(x)=m}{argmin} |x| \tag{22}$$

Das bleibt unter α-Robustheit bestehen, vorausgesetzt, α ist bekannt für ϕ, ψ: wir haben dann

$$\phi(m,p) = \underset{x:\psi(x,q)=m}{argmin} |x| \text{ f.a. } q : q \approx_\alpha p \tag{23}$$

Θ, das Lexikon

Wir kommen nun zum Kompressionsschema von Juba et al. Aber vorher müssen wir Θ besprechen.
Sie nehmen an, es gibt f.a. $m \in M$ eine *zufällige unendliche Folge*

$$\langle r_m^i \rangle_{i=1}^{\infty} \tag{24}$$

wobei

$$r_m^i \in \{0,1\}^i \tag{25}$$

sprich: jede Bedeutung hat unendlich viele Kodierungen, und zwar eine in *jeder* Länge.

Unser θ wäre also der "Bedeutungsgraph" mit Zuordnung

$$m \cong \{r_m^1, r_m^2, r_m^3, \ldots\} \tag{26}$$

und Θ die Menge aller dieser Zuordnungen.

Θ, das Lexikon

- Es gibt keine Beschränkung bezüglich Ambiguität und Synonymie:
- jedes r_m^i ist eine *rein zufällige* Folge der Länge i!
- Aber die Wahrscheinlichkeit, dass zwei m, m' die ersten i Kodierung identisch haben konvergiert natürlich schnell gegen 0 für $i \to \infty$!
- Man kann also davon ausgehen, dass alle Worte unterscheidbar sind

Das wichtige ist: θ ist geteilte Information zwischen Sprecher und Hörer, dementsprechend haben wir ϕ_θ passend zu ψ_θ.

Die Kodierung, intuitiv

Die Dekodierung ist nun einfach folgende (wir haben gesehen dass ψ wichtiger ist als ϕ):

Gegeben $p \in \Delta(M)$, $x \in \{0,1\}^*$ mit $|x| = i$, gibt ψ als Ausgabe:

dasjenige m: so dass

1. $r_m^i = x$
2. von allen m', die 1. erfüllen, hat m die *maximale Wahrscheinlichkeit* (unter p)

Die Kodierung muss nun so sein, dass die Dekodierung funktioniert!

Die Kodierung, formaler

Bekannt/geteilt sind, für ϕ, ψ,

1. θ
2. α

$\phi(m, p) =$ dasjenige r_m^i so dass i die kleinste Zahl ist, so dass, falls $m \neq m'$, $r_m^i = r_{m'}^i$, dann gilt $p(m) > \alpha^2 p(m')$.

$$\psi(x, q) = \underset{\{m : r_m^i = x\}}{argmax}\, q(m)$$

Die Kodierung, formaler

Diese Kodierung ist α-robust, formaler gesagt:

Lemma

Sei $p \approx_\alpha q$, $\theta \in \Theta$ wie oben definiert. Dann gilt f.a. $m \in M$

$$\psi(\phi(m, p), q) = m \tag{27}$$

D.h. wir haben α-Robustheit in diesem Schema!

Was uns als nächstes interessiert ist die (inverse) Kompression, d.h.:

Kompression

Für beliebiges $p \in \Delta(M)$, und für Θ, welches *uniform randomisiert* generiert wird, gilt

$$E_{m \sim p}(|\phi_\theta(m,p)|) \leq H(p) + 2 log \alpha + 2 \tag{28}$$

Damit man mit dieser Zahl etwas anfangen kann:

- Die Größe von α ist nicht sonderlich relevant, doppelt logarithmisch (z.B. $log_2(10) \approx 3.3$), $log_2(100) \approx 6.6$, $log_2(1000) \approx 9.9$)
- p spielt durchaus eine wichtige Rolle, und damit auch M. Wir gehen hier zwei Beispiele durch.

Kompression: Beispiele

Beispiel 1

Nimm an, $M = \{1, ..., 100\}$, und $p(m) = 1/100$ f.a. $m \in M$. Außerdem, $\alpha = 10$. Dann gilt:

$$H(p) = -100 \cdot 1/100 \cdot log(1/100) = -log(1/100) \approx 6.6 \tag{29}$$

$$2log\alpha = 2log10 = log100 = 6.6 \tag{30}$$

Also:

$$E_{m \sim p}(|\phi_\theta(m, p)|) \leq 6.6 + 6.6 + 2 = 15.2 \tag{31}$$

Das bedeutet: eine der 100 Bedeutungen wird durchschnittlich mit weniger als 15.2 Zeichen kodiert!

Kompression: Beispiele

Also:

$$E_{m\sim p}(|\phi_\theta(m,p)|) \leq 6.6 + 6.6 + 2 = 15.2 \tag{32}$$

Das bedeutet: eine der 100 Bedeutungen wird durchschnittlich mit weniger als 15.2 Zeichen kodiert!

Das ist relativ viel, wenn man bedenkt dass $log_2(100) = 6.6$, aber die "optimale Kodierung" hat eben keine α-Robustheit!

Beispiel 2

Nimm an,

- $M = \mathbb{N}$
- $p(m) = 1/2^m$ f.a. $m \in M$; also die **geometrische Verteilung**.

Außerdem, $\alpha = 100$. Dann gilt:

$$H(p) = -\frac{-(1/2log(1/2)) - 1/2log(1/2)}{1/2} = 2 \tag{33}$$

$$2log\alpha = 13.2 \tag{34}$$

Also:

$$E_{m\sim p}(|\phi_\theta(m,p)|) \leq 13.2 + 2 + 2 = 17.2 \tag{35}$$

Das bedeutet: eine der unendlich vielen Bedeutungen wird durchschnittlich mit weniger als 17.2 Zeichen kodiert!

Auch das ist relativ viel, wenn man bedenkt dass $H(p) = 2$, aber wir haben eben ein relativ großes α ($\alpha = 100$)!

Was kann man verbessern?

Wie wir gesehen haben, ist p ein Parameter der Kompression, hier gibt es also nichts zu holen. Ebenso α.

Wenn wir optimieren wollen, dann Θ! Denn bislang haben wir nur rein zufällige Kodierungen betrachtet; das geht aber bestimmt besser!

Man beachte aber: wir können nicht p benutzen, um Θ zu optimieren; das kann nur ϕ selber! Denn unsere Kodierung muss ja für jedes $p \in \Delta(M)$ funktionieren!

"Disambiguierung" des Lexikons

Hier liegt folgende Idee zugrunde: nimm an, r_m^i ist eindeutig, d.h.:

$$\text{falls } m \neq m', \text{ dann } r_m^i \neq r_{m'}^i$$

In diesem Fall haben wir eine eindeutige Kodierung von m der Länge i, und wir können alle längeren Kodierungen m *eliminieren*:

- Um m selbst zu kodieren, werden wir diese Kodierungen nie brauchen
- Aber evtl. werden diese Kodierungen die Kodierungen von *anderen Worten* negativ beeinflussen (also verlängern)!

"Disambiguierung" des Lexikons – formaler

Wir definiern $M_1 := M$, und

$$M_{i+1} = \{m \in M_i : \exists m' \in M_i \text{ s.d. } m' \neq m \& r^i_m = r^i_{m'}\} \tag{36}$$

M_{i+1} ist die Menge aller Bedeutungen, die keine eindeutige Kodierung der Länge $i+1$ haben. Nun definiere:

$$\hat{r}^i_m = \begin{cases} r^i_m & \text{falls } m \in M_i \\ -1 & \text{andernfalls} \end{cases} \tag{37}$$

-1 heisst: inexistent.

Ein Beispiel

Sei $M = \{a, b, c\}$, $p(m) = 1/3$ f.a. m. Ausserdem:

- $r_a^1 = 0$
- $r_b^1 = 1 = r_c^1$
- $r_a^2 = 00 = r_b^2$
- $r_c^2 = 01$

Unser altes Schema müsste b kodieren mit einem Wort $x : |x| > 2$; unser Schema mit r' kann b kodieren mit 00, denn a hat nur eine Kodierung!

Die geteilte Information

Eine Schwachstelle unseres alten Schemas ist es, dass wir selbst für endlich viele Bedeutungen *unendlich viel Information* zwischen Sprecher und Hörer teilen müssen.

Das ist in realistischen Szenarien nicht möglich, und es ist natürlich auch nicht nötig.

Im folgenden nehmen wir an, M ist endlich, und

$$\ell = \lceil log(|M|) \rceil + 1 \tag{38}$$

Jede Bedeutung m entspricht also einem Wort in $\{0,1\}^\ell$, oder alternativ einer Zahl $\leq 2^\ell$

Die geteilte Information

Sei π eine Primzahl in $[2^\ell, 2^{\ell+1})$ (die existiert)
Wir legen zwei zufällige Paramter $a \in \{1, ..., \pi - 1\}$, $b \in \{0, 1, ..., \pi - 1\}$ fest.
Jede Bedeutung m wird aufgefasst als eine Zahl in $m \in \{0, 1, ..., 2^\ell - 1\}$.

$$r_m^i = \begin{cases} ((am + b)(mod\pi))(mod2^i) & \text{falls } i \leq \ell \\ m & \text{falls } i = \ell + 1 \\ -1 & \text{falls } i > \ell + 1 \end{cases} \tag{39}$$

Ergebnisse

Das zentrale Ergebnis ist: die (inverse) Kompression ist, ebenso wie oben,

$$E_{m\sim p}(|\phi_\theta(m,p)|) \leq H(p) + 2log\alpha + 2 \tag{40}$$

Aber: die Menge der Information, die zwischen Hörer und Sprecher geteilt, ist nun begrenzt durch $log(|M|)$!

Wir haben also ein Szenario, das durchaus realistisch ist, im Sinne von:

- Wir haben ein robustes Schema zur effizienten Kodierung,
- mit einer Menge an geteilter Information, die durchaus vertretbar ist!

Ergebnisse

Wir haben also ein Szenario, das durchaus realistisch ist, im Sinne von:

- Wir haben ein robustes Schema zur effizienten Kodierung,
- mit einer Menge an geteilter Information, die durchaus vertretbar ist!

Was unrealistisch ist:

- Damit die Kommunikation *immer* erfolgreich ist, muss α bekannt sein – das ist sicher nicht immer der Fall in der Realität
- Aber: in der Realität ist Kommunikation ja auch nicht immer erfolgreich!

Zusammenfassung

Normale Kodierungsschemata sind entweder *universell* (unabhängig von Wahrscheinlichkeiten), oder *optimal* in Bezug ein eine Verteilung.

Juba et al. untersuchen einen dritten weg:

- Optimierung in Bezug auf Wahrscheinlichkeiten
- Robustheit durch ein fixes Lexikon

Tatsächlich kommt diese Art von Kodierung natürlicher Sprache sicher am Nächsten – jedenfalls nach meiner Intuition!