Retrievalmodelle

Probabilistische Modelle

Warum überhaupt Wahrscheinlichkeiten im IR?

In traditionellen IR-Systemen werden Antworten zu unpräzise Informationsbedürfnissen über eine (semantisch) unpräzise Abbildung auf Indexterme gefunden. Dieser Prozess beinhaltet eine Menge **Unschärfe** und **Unsicherheit**.

Die Wahrscheinlichkeitstheorie bietet geeignete Werkzeuge, um solche Unsicherheiten zu **quantifizieren**. Bereits sehr frühe Ansätze des IR greifen daher auf Methoden der Wahrscheinlichkeitstheorie zurück, um eine **optimale Rangfolge von Dokumenten** zu bestimmen.

Die grundlegende Idee ist es, Dokumente nach absteigender Relevanzwahrscheinlichkeit zu ordnen.

Das Probability Ranking Principle

Sei d ein Dokument aus der Kollektion. Die binäre Zufallsvariable R beschreibe die Relevanz eines Dokuments: R = 1 bedeutet also relevant, R = 0 bedeutet nicht relevant.

Wir müssen nun die Wahrscheinlichkeit P(R = 1|d,q) für die Relevanz von Dokument d für Anfrage q bestimmen. Wenn wir dies getan haben, geben wir Dokumente in **absteigender Relevanzwahrscheinlichkeit** zurück; das ist gerade das Prinzip des **Probability Ranking Principle** (PRP).

In der Praxis sortiert man nicht nach P(R = 1|d,q), sondern nach $\frac{P(R=1|d,q)}{P(R=0|d,q)}$, also nach dem Verhältnis der Wahrscheinlichkeiten für Relevanz und Nichtrelevanz (sog. **Gewinnchance** oder **Odds**).

Mit korrekt geschätzten Wahrscheinlichkeiten ist diese Ergebnisreihenfolge **optimal bezüglich der Ergebnisgüte** (unter bestimmten Annahmen an das Benutzerverhalten, z.B. muss die Relevanzwahrscheinlichkeit der einzelnen Dokumente unabhängig sein).

Binary Independence Model

Wir betrachten hier zur Vereinfachung wie im Booleschen Modell Dokumente als **Spaltenvektoren der Inzidenzmatrix**. Ein Dokument d wird also repräsentiert als $\mathbf{d}=(w_1,\ldots,w_{|V|})^T$, wobei $w_i=1$ falls d Term i enthält und 0 sonst. Die Anfrage ist eine Menge von Termen, also $q \subseteq V$.

Die Ähnlichkeit eines Dokuments d und einer Anfrage q ist nun

$$sim(q,\boldsymbol{d}) := \frac{P(R=1|\boldsymbol{d},q)}{P(R=0|\boldsymbol{d},q)}$$

was wir mit Bayes‘ Regel umwandeln können in

$$sim(q,\boldsymbol{d}) := \frac{P(\boldsymbol{d}|R=1,q)\cdot P(R=1|q)}{P(\boldsymbol{d}|R=0,q)\cdot P(R=0|q)}$$

Dabei sind

- $P(\mathbf{d}|R=1,q)$ die Wahrscheinlichkeit, dass ein zufällig gewähltes Dokument, das relevant für q ist, gerade die Repräsentation **d** hat,
- $P(R=1|q)$ die Wahrscheinlichkeit, dass ein zufällig gewähltes Dokument relevant für q ist.

Binary Independence Model

Da P(R = 1|q) und P(R = 0|q) für alle Dokumente der Kollektion identisch sind, spielen sie hier keine Rolle, also:

$$sim(q, \boldsymbol{d}) \propto \frac{P(\boldsymbol{d}|R = 1, q)}{P(\boldsymbol{d}|R = 0, q)}$$

Die zentrale Annahme des **Binary Independence Models** ist nun, dass die Termvorkommen in Dokument d **unabhängig** voneinander sind, wir also die Relevanzwahrscheinlichkeiten wiefolgt zerlegen können:

$$sim(q, \boldsymbol{d}) \propto \frac{\prod_{k_i|w_i=1} P(k_i|R = 1, q) \cdot \prod_{k_i|w_i=0} P(\overline{k_i}|R = 1, q)}{\prod_{k_i|w_i=1} P(k_i|R = 0, q) \cdot \prod_{k_i|w_i=0} P(\overline{k_i}|R = 0, q)}$$

Dabei ist $P(k_i|R = 1, q)$ die Wahrscheinlichkeit, dass der Indexterm k_i in einem relevanten Dokument auftritt, und entsprechend $P(\overline{k_i}|R = 1, q)$ die Wahrscheinlichkeit, dass k_i nicht in einem relevanten Dokument auftritt.

Binary Independence Model

Zur Vereinfachung schreiben wir

$$p_i := P(k_i|R = 1, q)$$
$$q_i := P(k_i|R = 0, q)$$

Weil außerdem $P(k_i|R = 1, q) + P(\overline{k_i}|R = 1, q) = 1$ (analog für R=0), können wir schreiben

$$sim(q, \boldsymbol{d}) \propto \frac{\prod_{k_i|w_i=1} p_i \cdot \prod_{k_i|w_i=0}(1 - p_i)}{\prod_{k_i|w_i=1} q_i \cdot \prod_{k_i|w_i=0}(1 - q_i)}$$

Wir nehmen nun den Logarithmus, was nur die absoluten Werte, aber nicht die Reihenfolge verändert, und erhalten

$$sim(q, \boldsymbol{d}) \propto \log \prod_{k_i|w_i=1} p_i + \log \prod_{k_i|w_i=0} (1 - p_i)$$
$$- \log \prod_{k_i|w_i=1} q_i - \log \prod_{k_i|w_i=0}(1 - q_i)$$

Der Logarithmus sorgt dafür, dass die Werte nicht zu klein werden, sonst könnten wir an die Grenze des darstellbaren Zahlenbereichs kommen.

Binary Independence Model

Wir fügen nun Ausdrücke hinzu, die sich gegenseitig auslöschen:

$$sim(q, \boldsymbol{d}) \propto \log \prod_{k_i | w_i = 1} p_i + \log \prod_{k_i | w_i = 0} (1 - p_i)$$

$$+ \log \prod_{k_i | w_i = 1} (1 - p_i) - \log \prod_{k_i | w_i = 1} (1 - p_i)$$

$$- \log \prod_{k_i | w_i = 1} q_i - \log \prod_{k_i | w_i = 0} (1 - q_i)$$

$$+ \log \prod_{k_i | w_i = 1} (1 - q_i) - \log \prod_{k_i | w_i = 1} (1 - q_i)$$

Binary Independence Model

Wir erhalten durch Umformen

$$sim(q, \boldsymbol{d}) \propto \log \prod_{k_i | w_i = 1} \frac{p_i}{1 - p_i} + \log \prod_{k_i} (1 - p_i) + \log \prod_{k_i | w_i = 1} \frac{1 - q_i}{q_i} - \log \prod_{k_i} (1 - q_i)$$

Binary Independence Model

Nehmen wir nun an, dass $p_i = q_i$ für alle Terme k_i, die nicht in der Anfrage vorkommen, und wandeln wir die Logarithmen der Produkte in Summen von Logarithmen um, erhalten wir schließlich

$$sim(q, \boldsymbol{d}) \propto \sum_{k_i \in q \wedge w_i = 1} \log \frac{p_i}{1 - p_i} + \log \frac{1 - q_i}{q_i}$$

Dies ist die wesentliche Rankingformel für das probabilistische Retrievalmodell.

Aber: Da wir die Menge der relevanten Dokumente natürlich nicht kennen, müssen wir noch festlegen, wie wir p_i und q_i bestimmen.

Schätzen von Wahrscheinlichkeiten im BIM

Wir verwenden dazu folgende Symbole:

- N ist die Anzahl der Dokumente
- n_i ist die Anzahl der Dokumente, die Term k_i enthalten
- R ist die Anzahl der Dokumente, die für Anfrage q relevant sind
- r_i ist die Anzahl der relevanten Dokumente, die Term k_i enthalten

Damit können wir nun folgende Kontingenztafel aufstellen:

	relevant	nichtrelevant	gesamt
Dok., die k_i enthalten	r_i	$n_i - r_i$	n_i
Dok., die k_i nicht enthalten	$R - r_i$	$N - n_i - (R - r_i)$	$N - n_i$
Alle Dokumente	R	$N - R$	N

Wäre diese Information für eine Anfrage verfügbar, so ergäbe sich

$$p_i = \frac{r_i}{R}, q_i = \frac{n_i - r_i}{N - R}$$

Schätzen von Wahrscheinlichkeiten im BIM

Wir können jedenfalls unsere Ähnlichkeit jetzt wiefolgt aufschreiben:

$$sim(q, \boldsymbol{d}) \propto \sum_{k_i \in q \wedge w_i = 1} \log \frac{r_i (N - n_i - R + r_i)}{(R - r_i)(n_i - r_i)}$$

Hinzufügen eines Glättungsterms ergibt

$$sim(q, \boldsymbol{d}) \propto \sum_{k_i \in q \wedge w_i = 1} \log \frac{(r_i + 0{,}5)(N - n_i - R + r_i + 0{,}5)}{(R - r_i + 0{,}5)(n_i - r_i + 0{,}5)}$$

Weil wir initial keine Informationen über relevante Dokumente haben, setzen wir R = r_i = 0 und erhalten

$$sim(q, \boldsymbol{d}) \propto \sum_{k_i \in q \wedge w_i = 1} \log \frac{N - n_i + 0{,}5}{n_i + 0{,}5}$$

Das ist gerade eine mögliche Form der IDF; oft lässt man auch das n_i im Zähler weg, um Anomalien zu vermeiden, und erhält so praktisch die IDF-Formulierung, die wir vorher verwendet haben.

Schätzen von Wahrscheinlichkeiten im BIM

Um die Wahrscheinlichkeiten besser zu schätzen, braucht man Relevanzinformation

- durch **explizites Feedback** des Benutzers,
- durch abgeleitetes Feedback, z.B. aus **Click- und Browseverhalten**, oder
- indem man annimmt, dass die besten k Dokumente relevant sind (**Blind Feedback**).

Man kann dann die so gewonnenen relevanten Dokumente verwenden, um die Wahrscheinlichkeiten (ggf. wieder mit Glättung) genauer zu schätzen, und so die Qualität des Rankings zu verbessern, potentiell in mehreren Schritten.

Das Binary Independence Model verwendet keinerlei Information über die Termhäufigkeit, was sein wesentlicher Nachteil neben der Notwendigkeit von Relevanzinformation ist.

Okapi BM25

Die Rankingformel des **Okapi-Systems** greift die Probleme des probabilistischen Retrievalmodells auf und erweitert das Modell um

- eine Komponente, die die Termfrequenz berücksichtigt und
- eine Komponente, die den Einfluss der Dokumentlänge berücksichtigt (**Dokumentlängennormalisierung**).

Die Formeln aus der Familie der **BM-Modelle** (für Best Match) wurden dabei heuristisch weiterentwickelt, beginnend mit der BM15-Formel, die ein gedämpftes Termgewicht zusammen mit der IDF-Formulierung aus BIM (die alleine die BM1-Formel darstellt) einsetzt:

$$score_{BM15}(q,d) := \sum_{t \in q} \frac{(K+1)tf_{t,d}}{K + tf_{t,d}} \cdot idf_t$$

Einfluss der Dokumentlänge

Offensichtlich sind große $tf_{t,d}$-Werte wahrscheinlicher in langen Dokumenten, die auf diese Art einen "Vorteil" bei der Scorebestimmung haben.

Warum könnten Dokumente länger sein?

- **Ausführlichkeit**: Dann wäre der beobachtete tf-Wert höher als gerechtfertigt
- **größere inhaltliche Breite**: Dann wäre ein hoher tf-Wert gerechtfertigt

In einem realen Dokument finden wir wahrscheinlich beide Effekte. Eine gewisse Längennormalisierung erscheint also sinnvoll.

Einfluss der Dokumentlänge

Die **Dokumentlänge** eines Dokuments d ist die Summe seiner Termvorkommen:

$$dl_d := \sum_{t \in V} tf_{t,d}$$

Wir betrachten außerdem die **mittlere Dokumentlänge**:

$$avgdl := \frac{\sum_{d \in D} dl_d}{|D|}$$

Wir werden unseren Score um eine Komponente zur **Längennormalisierung** erweitern:

$$B := \left((1 - b) + b \frac{dl_d}{avgdl} \right)$$

Hierbei steuert der Parameter b den Einfluss der Längennormalisierung.

BM25-Formel

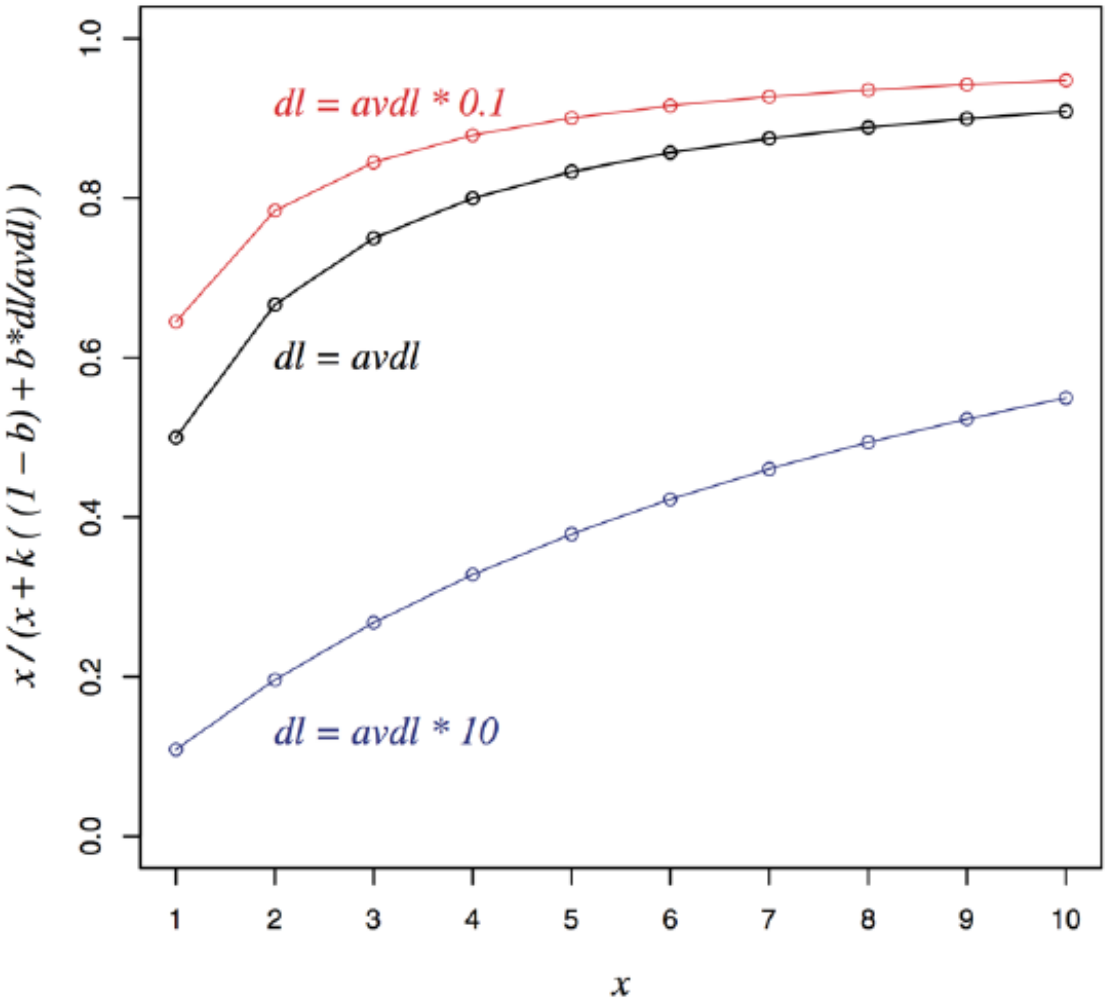
Zusammen mit der Dämpfung der Termfrequenz ergibt sich dann die folgende Formel, die als **BM25-Formel** bekannt ist (setze $tf' := \frac{tf}{B}$ und vereinfache):

$$score_{BM25}(q,d) = \sum_{t \in q} \frac{(k+1)tf_{t,d}}{k\left((1-b)+b\frac{dl_d}{avgdl}\right)+tf_{t,d}} \cdot idf_t$$

Typische Startwerte für die Parameteroptimierung sind k∈[1.2;2.0] und b=0.75.

Einfluss der Dokumentlänge

Diese Abbildung zeigt noch einmal den Einfluss der Dokumentlängennormalisierung und der Dämpfung der Termfrequenz auf den Beitrag der Termfrequenz zum Score:



Der Okapi-BM25-Score

Der **Okapi-BM25-Score** schlägt in Benchmarks Systeme, die auf tf·idf basieren, in der Regel deutlich.

Weil auch die tf-Komponente des BM25-Modells außerdem durch probabilistische Argumente begründet werden kann, ist es fundierter und weniger heuristisch als das tf·idf-Modell, das wir im Vektorraummodell verwendet haben.

Retrievalmodelle

Generative Sprachmodelle

Statistische Sprachmodelle (Language Models)

Statistische Sprachmodelle spielen in vielen Anwendungsbereichen eine Rolle, die mit natürlicher Sprache arbeiten, z.B. maschinelles Übersetzen, Spracherkennung, Sprachanalyse, und natürlich auch im Information Retrieval. Wir haben das Konzept bereits bei der kontext-sensitiven Rechtschreibkorrektur verwendet.

Ein Sprachmodell modelliert die in einer Sprache auftretenden Sätze **statistisch**. Es erlaubt, die **Wahrscheinlichkeit** zu bestimmen, mit der eine vorgegebene Wortfolge vorkommt. Ein Sprachmodell wird anhand vieler **Beispielsätze** statistisch gelernt. Es verwendet also gerade keine generierende Grammatik; diese wäre in der Regel auch viel zu komplex und könnte nicht bestimmt werden.

Statistische Sprachmodelle

Wir möchten also die Wahrscheinlichkeit bestimmen, mit der die Wortsequenz w1…wn auftritt; man sagt auch, die Wahrscheinlichkeit, mit der das Modell die Wortsequenz **generiert**. Mit den üblichen Rechenregeln für Wahrscheinlichkeiten erhalten wir

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

Die Wahrscheinlichkeit, mit der ein Wort in der Folge auftritt, hängt also von **allen Wörtern** ab, die davor in der Folge aufgetreten sind. Solche komplexen Abhängigkeiten kann man praktisch nicht bestimmen, wir verwenden daher vereinfachende Approximationen.

Die am häufigsten eingesetzten Sprachmodelle verwenden **Unigramme** und **Bigramme**.

Unigramm- und Bigramm-Sprachmodelle

Bei einem **Unigramm-Sprachmodell** wird die Wahrscheinlichkeit einer Wortsequenz auf die Wahrscheinlichkeit der einzelnen Wörter zurückgeführt:

$$P(w_1 w_2 \ldots w_n) = P(w_1) \cdot P(w_2) \cdot \cdots \cdot P(w_n)$$

Dieses Modell berücksichtigt die Reihenfolge der Wörter nicht, es betrachtet also nur die Wortsequenz als Bag-of-Words.

Ein **Bigramm-Sprachmodell** konditioniert dagegen die Teilwahrscheinlichkeiten mit dem vorhergehenden Wort:

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

Das ist gerade das Modell, das wir für die Fehlerkorrektur verwendet haben.

Unigramm- und Bigramm-Sprachmodelle

Man schätzt die jeweiligen Wahrscheinlichkeiten auf Basis der **Vorkommen der Wörter** (bzw. der Wortpaare bei Bigramm-Modellen) **in einer großen Menge von Dokumenten**, die charakteristisch für die jeweilige Sprache sind.

$$P(w_i) = \frac{\#\text{ Vorkommen von } w_i}{\sum_k \#\text{ Vorkommen von } w_k}$$

$$P(w_i|w_k) = \frac{\#\text{ Vorkommen von } (w_k, w_i)}{\#\text{ Vorkommen von } w_k}$$

Beispiel: Unigramm-Sprachmodell

Nehmen wir an, dass wir folgende Unigramm-Wahrscheinlichkeiten geschätzt haben:

P(Trier)=0,01	P(Passau)=0,01	P(Saarbrücken)=0,01
P(ist)=0,1	P(eine)=0,1	P(schöne)=0,02
P(Stadt)=0,05	P(in)=0,1	P(Bayern)=0,05
P(Saarland)=0,05	P(ein)=0,1	P(der)=0,1
P(die)=0,1	P(das)=0,1	P(und)=0,1

Dann erhalten wir folgende Sequenzwahrscheinlichkeiten:

P(Passau ist in Bayern)	$= 0{,}01 \cdot 0{,}1 \cdot 0{,}1 \cdot 0{,}05$
	$= 0{,}00005$
P(Passau ist eine Stadt in Bayern)	$= 0{,}01 \cdot 0{,}1 \cdot 0{,}1 \cdot 0{,}05 \cdot 0{,}1 \cdot 0{,}05$
	$= 0{,}000000025$

Da das Modell die Bedeutung und die Struktur der Sprache nicht versteht, kann es auch “sinnlose” Sequenzen wie “Passau Bayern und eine Saarbrücken ist” generieren.

Sprachmodelle im Information Retrieval

Im Information Retrieval müssen wir **Relevanzwahrscheinlichkeiten P(R = 1|q,d)** schätzen; da die Anfrage q die "Eingabe" der Schätzung ist, könnten wir stattdessen auch **P(d|q)** schreiben – die Wahrscheinlichkeit, dass Dokument d geliefert wird, wenn zufällig ein für Anfrage q relevantes Dokument gewählt wird.

Jetzt können wir die Bayes'sche Regel anwenden:

$$P(d|q) = \frac{P(q|d) \cdot P(d)}{P(q)}$$

P(q) ist für eine feste Anfrage konstant, spielt also für das Ranking keine Rolle. P(d) könnte man verwenden, um "guten" Dokumenten einen höhere Wahrscheinlichkeit zu geben (**document prior**); hat man keine Informationen über die Dokumentqualität, nimmt man Gleichverteilung an und kann diesen Faktor für das Ranking ebenfalls ignorieren.

Wir erhalten also das folgende **Query-Likelihood-Modell**:

$$P(d|q) \propto P(q|d)$$

Sprachmodelle im Information Retrieval

Wir können also die Relevanzwahrscheinlichkeit von Dokument d schätzen, indem wir ein Sprachmodell P(.|d) für d ermitteln und dann schätzen, mit welcher Wahrscheinlichkeit die **Anfrage q von diesem Sprachmodell generiert** wird.

Für eine Anfrage q = {t1,…,tn} und ein Unigramm-Sprachmodell für Dokument d können wir also P(d|q) schätzen als:

$$P(d|q) \propto n! \cdot \prod_{i=1}^{n} P(t_i|d)$$

$$\propto \sum_{i=1}^{n} \log P(t_i|d)$$

Sprachmodelle im Information Retrieval

Der Faktor n! berücksichtigt dabei alle Möglichkeiten, die Terme aus q als Sequenz darzustellen; er spielt für das Ranking keine Rolle, da er für alle Dokumente gleich ist. Wir ordnen die Dokumente in absteigender Relevanzwahrscheinlichkeit.

Die Anfrage kann auch als Multimenge formuliert werden, so dass ein Term mehrfach vorkommen kann. Um ein Bigramm-Modell zu verwenden, muss die Anfrage als Sequenz von Wörtern gestellt werden.

Beispiel

Wir betrachten die Anfrage **q={Passau, Bayern}** und das Sprachmodell von eben, das für Dokument d1 berechnet worden sei. Wir erhalten

$P(q|d1) \propto \log 0{,}01 + \log 0{,}05 = -2 - 1{,}3 = -3{,}3$

Für zwei andere Dokumente d2 und d3 könnten wir folgende Sprachmodelle bestimmt haben (Ausschnitte):

$P(\text{Passau}|d2)=0{,}001$ $P(\text{Passau}|d3)=0{,}1$

$P(\text{Bayern}|d2)=0{,}01$ $P(\text{Bayern}|d3)=0$

Wir erhalten also

$P(q|d2) \propto \log 0{,}001 + \log 0{,}01 = -3 - 2 = -5$

Dokument d2 hat also eine geringere Relevanzwahrscheinlichkeit als d1.

Wir können P(q|d3) dagegen nicht auf diesem Weg bestimmen, da P(Bayern|d3) = 0 und somit der Logarithmus undefiniert ist. Das Modell sagt also, dass d3 **unmöglich relevant für q** sein kann, weil “Bayern” nicht in d3 vorkommt.

Glättung von Sprachmodellen

Die bisher diskutierten Sprachmodelle verwenden eine **konjunktive** Semantik: Alle Anfrageterme müssen im Dokument vorkommen (d.h. eine positive Generierungswahrscheinlichkeit haben), damit ein Dokument überhaupt relevant sein kann.

Für viele Anwendungen ist dies zu strikt. Man verwendet daher **Glättungs-** oder **Smoothing-Methoden**, um auch im Fall fehlender Terme eine gewisse Wahrscheinlichkeit berücksichtigen zu können.

Man verwendet dazu ein **Hintergrund-Sprachmodell**, das auf der ganzen Kollektion geschätzt wird und das man mit dem Sprachmodell des Dokuments mischt.

Glättung von Sprachmodellen

Wir benötigen dazu die Gesamtlänge T der Kollektion:

$$T := \sum_{t \in V} cf_t$$

Das Hintergrund-Sprachmodell (oder Kollektionsmodell) P_c berechnen wir nun für jeden Term als

$$P_c(t) := \frac{cf_t}{T}$$

Wir können nun das Sprachmodell eines Dokuments d **linear** mit dem Hintergrundmodell **interpolieren** (auch als **Jelinek-Mercer-Smoothing** bekannt):

$$P_{JM}(t|d) := \lambda P(t|d) + (1 - \lambda) P_c(t)$$

Der Parameter wird dabei durch Tuning bestimmt.

Beispiel

Wir betrachten wieder die Dokumente und die Anfrage von eben, glätten aber die Modelle mit λ=0,5 und folgendem Hintergrundmodell:

$P_c(\text{Passau})=0{,}001$ $P_c(\text{Bayern})=0{,}01$

Wir erhalten folgende geglätteten Sprachmodelle:

$P_{LM}(\text{Passau}|d1) = 0{,}5\cdot0{,}01 + 0{,}5\cdot0{,}001 = 0{,}0055$

$P_{LM}(\text{Passau}|d2) = 0{,}5\cdot0{,}001 + 0{,}5\cdot0{,}001 = 0{,}001$

$P_{LM}(\text{Passau}|d3) = 0{,}5\cdot0{,}1 + 0{,}5\cdot0{,}001 = 0{,}0505$

$P_{LM}(\text{Bayern}|d1) = 0{,}5\cdot0{,}01 + 0{,}5\cdot0{,}01 = 0{,}01$

$P_{LM}(\text{Bayern}|d2) = 0{,}5\cdot0{,}01 + 0{,}5\cdot0{,}01 = 0{,}01$

$P_{LM}(\text{Bayern}|d3) = 0{,}5\cdot0 + 0{,}5\cdot0{,}01 = 0{,}005$

und damit

$P_{LM}(q|d1)\propto-4{,}26$ $P_{LM}(q|d2)\propto-5$ $P_{LM}(q|d3)\propto-3{,}58$

Abschließende Bemerkungen zu Sprachmodellen

Während bei tf·idf und BM25 ein in der Kollektion **seltener Term** wichtiger als ein häufiger, ist es in Sprachmodellen umgekehrt: Im Hintergrundmodell haben in der Kollektion **häufige Terme** eine höhere Wahrscheinlichkeit als seltene Terme.

Neben dem hier gezeigten Query-Likelihood-Modell gibt es noch andere Ansätze, die Sprachmodelle verwenden, z.B. das Document-Likelihood-Modell oder die Divergenz von Anfrage- und Dokumentmodell.

Sprachmodelle erzielen in den meisten Benchmarks **bessere Ergebnisse** als andere Modelle, z.B. BM25.