Chapter IR:III

III. Retrieval Models

- Overview of Retrieval Models
- Boolean Retrieval
- Vector Space Model
- Binary Independence Model
- Okapi BM25
- Divergence From Randomness
- Latent Semantic Indexing
- Explicit Semantic Analysis
- Language Models
- Combining Evidence
- Learning to Rank

Language Models

Background

Language models in general include methods to represent the syntactical structures of languages to study them, and to solve natural language processing tasks.

A key goal of modeling a language is to solve the membership problem:
Given a string and a language, decide whether the string belongs to the language.

Two complementary approaches have been pursued:

- Formal languages
  Theoretical approach with an explicit grammar specification and applications in comparably small, controlled languages (e.g., query languages, programming languages).
- Statistical language models
  Probabilistic approach where grammar is captured only implicitly by analyzing large text collections. Can be applied in less controlled situations.

Important applications of statistical language models:

- Part-of-speech tagging
- Machine translation
- Speech and handwriting recognition
- Information retrieval

Language Models

Basics: Grammar

- Alphabet Σ.

  An alphabet Σ is a non-empty set of signs or symbols.

- Word w.

  A word w is a finite sequence of symbols from Σ. The length of a word $|w|$ is the number of symbols it is made of.

  ε denotes the empty word; it is the only word of length 0.
  Σ^* denotes the set of all words over Σ.

- Language L.

  A language L is a set of words over an alphabet Σ.

- Grammar G.

  A grammar G is a calculus to define a language—and a set of rules by which words can be derived. The language corresponding to G contains all words that can be generated using its rules.

Language Models

Basics: Grammar

- Alphabet Σ.

  An alphabet Σ is a non-empty set of signs or symbols.

- Word w.

  A word w is a finite sequence of symbols from Σ. The length of a word $|w|$ is the number of symbols it is made of.

  ε denotes the empty word; it is the only word of length 0.
  Σ^* denotes the set of all words over Σ.

- Language L.

  A language L is a set of words over an alphabet Σ.

- Grammar G.

  A grammar G is a calculus to define a language—and a set of rules by which words can be derived. The language corresponding to G contains all words that can be generated using its rules.

Remarks [Kastens 2005] :

- Language properties can be defined at different levels of abstraction. Level 1 deals with the notation of basic symbols, Level 2 deals with syntactical structures of a language.
  To distinguish the levels of a grammar under consideration, the following terminology is used:
  - Level 1: alphabet, character, word, language
  - Level 2: vocabulary, symbol, sentence, language
- The words {alphabet, vocabulary}, {character, symbol}, and {word, sentence} correspond to one another at the two respective levels.

Language Models

Basics: Grammar (continued)

Definition 2 (Grammar)

A grammar is a 4-tuple $G = (N, \Sigma, P, S)$, where

- N is a finite set of nonterminal symbols
- Σ is a finite set of terminal symbols, $N \cap \Sigma = \emptyset$
- P is a finite set of (production) rules

$$P \subset (N \cup \Sigma)^* \, N \, (N \cup \Sigma)^* \times (N \cup \Sigma)^*$$

- S is a start symbol, $S \in N$

Remarks:

- A production rule comprises a left side (premise) and a right side (conclusion), which are both words composed of terminals and nonterminals. The left side must contain at least one nonterminal and the right side may, unlike the left side, be the empty word. [Wikipedia]
- A rule can be applied to a word composed of terminals and nonterminals, where an arbitrary occurrence of the left side of the rule is replaced with its right side: $w \to w'$.
- Given the rule $w \to w'$, the words w, w' are in a *transition relation*. A sequence of rule applications is called a *derivation*.

Language Models

Basics: Grammar (continued)

Definition 3 (Generated Language)

Given a grammar $G = (N, \Sigma, P, S)$, it generates the language $L(G)$ which contains exactly the words which are composed of only terminal symbols and which can be derived in a finite number of steps:

$$L(G) := \{w \in \Sigma^* \mid S \to_G^* w\},$$

where $\to_G^*$ denotes the arbitrary application of rules in G, i.e., the reflexive transitive closure of the transition relation $\to_G$.

Language Models

Basics: Grammar (continued)

Definition 3 (Generated Language)

Given a grammar $G = (N, \Sigma, P, S)$, it generates the language $L(G)$ which contains exactly the words which are composed of only terminal symbols and which can be derived in a finite number of steps:

$$L(G) := \{w \in \Sigma^* \mid S \to_G^* w\},$$

where $\to_G^*$ denotes the arbitrary application of rules in G, i.e., the reflexive transitive closure of the transition relation $\to_G$.

Example:

$G = (N, \Sigma, P, S)$ where $N = \{S, A, B\}$, $\Sigma = \{a, b\}$ and the following rules P:

$$\begin{array}{lcl} S & \to & ABS \\ S & \to & \varepsilon \end{array} \qquad \begin{array}{lcl} BA & \to & AB \\ BS & \to & b \\ Bb & \to & bb \\ Ab & \to & ab \\ Aa & \to & aa \end{array}$$

Remarks:

- Conventionally, nonterminal symbols are denoted with uppercase letters and terminal symbols with lowercase letters.
- A given language can be generated by an arbitrary number of grammars.
- Another grammar that generates the exemplified languages is:
  $N = \{S, A, B\}$, $\Sigma = \{a, b\}$, $P = \{S \to aSb, S \to \varepsilon\}$

Language Models

Basics: Chomsky Hierarchy

Grammars are organized into four classes in terms of the complexity of the languages they generate.

- Type 0.
  The production rules in P are unrestricted.
- Type 1 $\sim$ context-sensitive.
  For all rules $w \to w' \in P$ holds: $|w| \leq |w'|$
- Type 2 $\sim$ context-free.
  For all rules $w \to w' \in P$ holds: w is a single variable; i.e., $w \in N$.
- Type 3 $\sim$ regular.
  In addition to being a Type 2 grammar, for all rules $w \to w' \in P$ holds: $w' \in (\Sigma \cup \Sigma N)$, i.e., the right side of the rules either are a terminal symbol or a terminal symbol followed by nonterminal.

Language Models

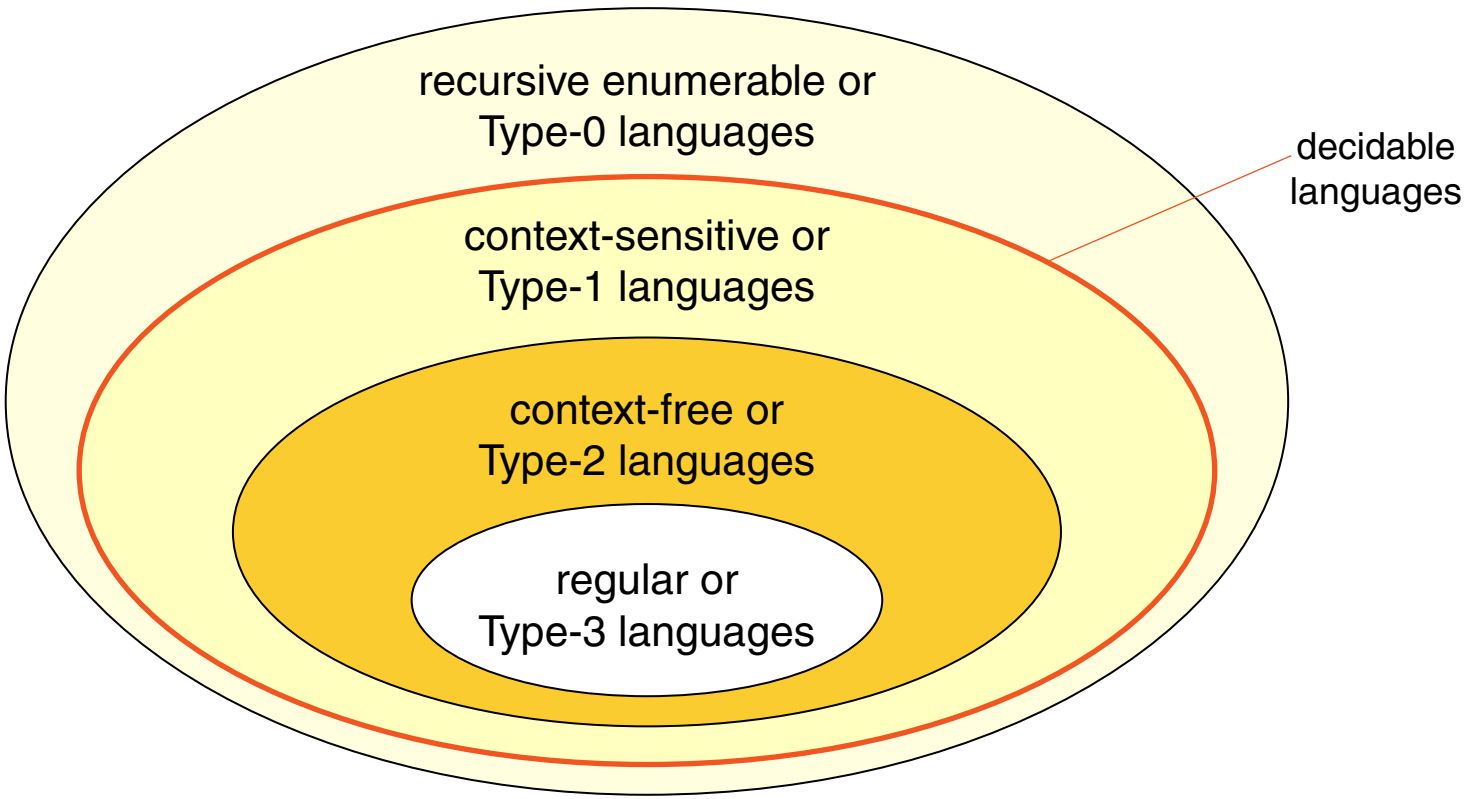
Basics: Chomsky Hierarchy

Grammars are organized into four classes in terms of the complexity of the languages they generate.

- Type 0.
  The production rules in P are unrestricted.
- Type 1 $\sim$ context-sensitive.
  For all rules $w \to w' \in P$ holds: $|w| \leq |w'|$
- Type 2 $\sim$ context-free.
  For all rules $w \to w' \in P$ holds: w is a single variable; i.e., $w \in N$.
- Type 3 $\sim$ regular.
  In addition to being a Type 2 grammar, for all rules $w \to w' \in P$ holds: $w' \in (\Sigma \cup \Sigma N)$, i.e., the right side of the rules either are a terminal symbol or a terminal symbol followed by nonterminal.

Language Models

Basics: Chomsky Hierarchy (continued)

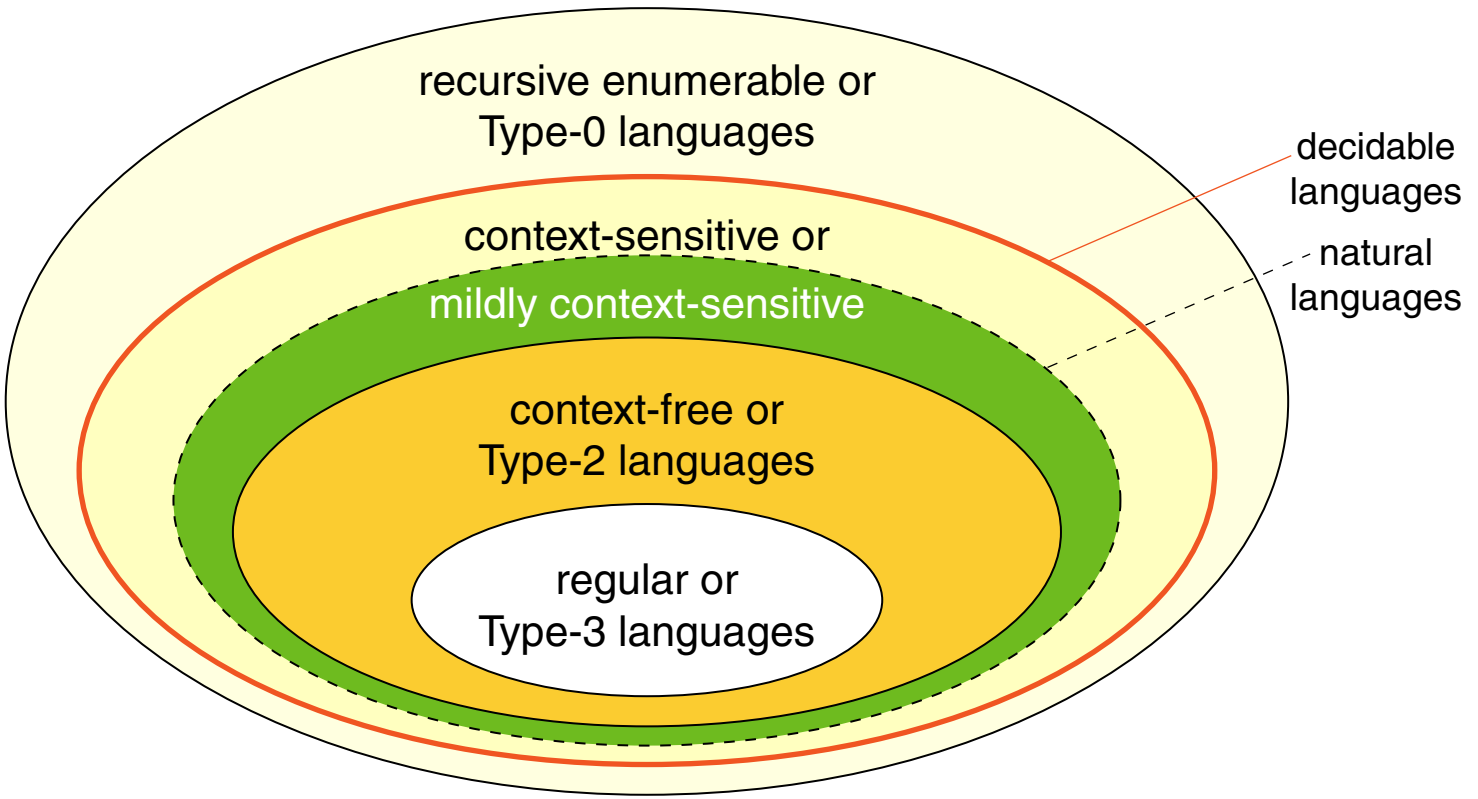


Definition 4 (Language of Type)

A language $L \subseteq \Sigma^*$ is called a language of Type 0 (Type 1, Type 2, Type 3), if a grammar G of Type 0 (Type 1, Type 2, Type 3) exists, so that $L(G) = L$.

Language Models

Basics: Chomsky Hierarchy (continued)



A grammar G is called mildly context-sensitive if

- it generates the copy language $\{ww \mid w \in \Sigma^*\}$ with $\Sigma = \{a, b\}$,
- the length difference of two strings from the length-ordered $L(G)$ is bounded by a constant,
- the membership problem is decidable in polynomial time.

Remarks:

- The Chomsky hierarchy is a hierarchy with true subset relationships: Type 3 $\subset$ Type 2 $\subset$ Type 1 $\subset$ Type 0.
- All languages of Type 1, 2, and 3 are decidable:
  i.e., the membership problem is decidable for all languages of Type 1, 2, and 3;
  i.e., an algorithm exists, which, given a grammar G and a word w, decides in finite time if $w \in L(G)$ holds.
- The set of Type 0 languages is identical to the set of recursively enumerable or semidecidable languages. Thus, Type 0 languages exist, which are undecidable.
- Languages and grammars of Types 2 and 3 play a central role in compiler design, namely syntactic analysis (Type 2), and lexical analysis and tokenization (Type 3), respectively.
- Chomsky introduced context-sensitive grammars as a way to describe the syntax of natural language where it is often the case that a word may or may not be appropriate in a certain place depending on the context.
  Although certain features of languages (e.g., cross-serial dependencies) are not context-free, it is an open question how much of the expressive power of context-sensitive grammars is needed to capture the context-sensitivity found in natural languages. Subsequent research in this area has focused on the more computationally tractable mildly context-sensitive languages. [Wikipedia]
- Despite a considerable amount of work on the subject, there is no generally accepted formal definition of mild context-sensitivity.

Language Models

Basics: Calculi

	Calculi for Language Generation
Type 0	unrestricted grammar generic Turing machine
Type 1	context-sensitive grammar linear bounded Turing machine [Wikipedia]
Type 2	context-free grammar push down automaton
Type 3	regular grammar (non)deterministic finite automaton regular expression

Language Models

Basics: Calculi

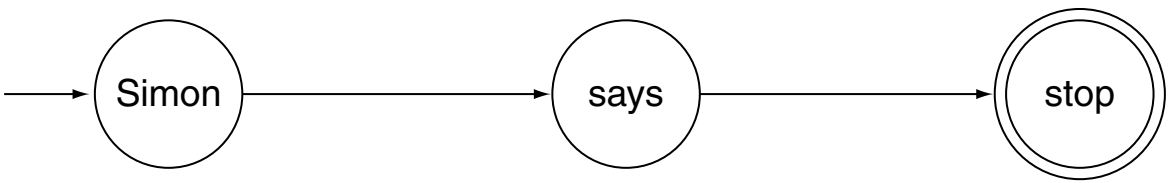
	Calculi for Language Generation
Type 0	unrestricted grammar generic Turing machine
Type 1	context-sensitive grammar linear bounded Turing machine [Wikipedia]
Type 2	context-free grammar push down automaton
Type 3	regular grammar (non)deterministic finite automaton regular expression

	Complexity of the Membership Problem
Type 0	undecidable
Type 1	exponential complexity, NP-hard
Type 2	$O(n^3)$
Type 3	linear complexity

Language Models

Example: Deterministic Language Model

Grammar G_1 as deterministic finite automaton:



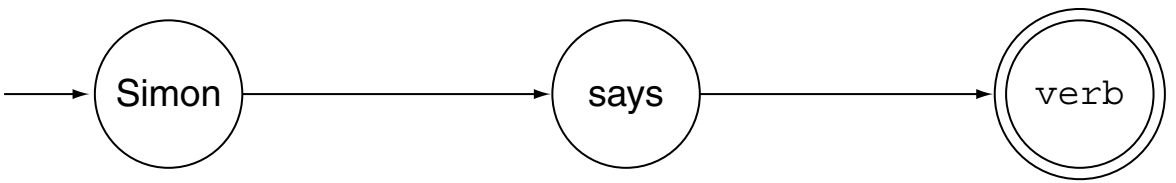
Generated language:

- $L(G_1) = \{\texttt{Simon says stop}\}$
- How to allow for other “Simon says” sentences?

Language Models

Example: Deterministic Language Model

Grammar G_2 as deterministic finite automaton:



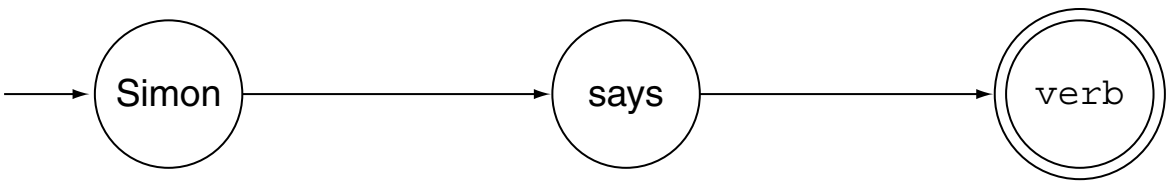
Generated language:

- Let $\texttt{verb} = \{\texttt{jump}, \texttt{run}, ...\}$ denote the set of all verbs.
- $L(G_2)$ contains `Simon says` sentences, e.g.:
  `Simon says jump`, `Simon says run`, ...
- $|L(G_2)| = |\texttt{verb}|$

Language Models

Example: Deterministic Language Model

Grammar G_2 as deterministic finite automaton:



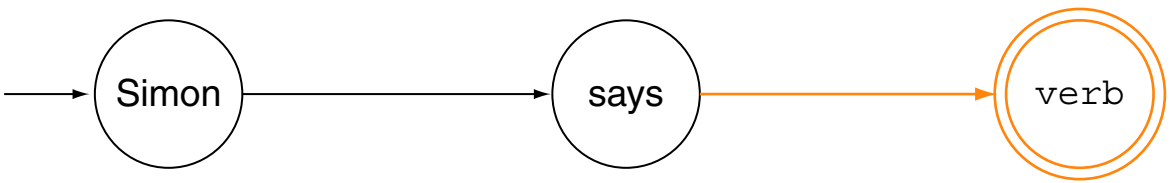
Generated language:

- Let $\texttt{verb} = \{\texttt{jump}, \texttt{run}, ...\}$ denote the set of all verbs.
- $L(G_2)$ contains `Simon says` sentences, e.g.:
  `Simon says jump`, `Simon says run`, ...
- $|L(G_2)| = |\texttt{verb}|$
- Is the sentence `Simon says science` member of $L(G_2)$?

Language Models

Example: Deterministic Language Model

Grammar G_2 as deterministic finite automaton:



Generated language:

- Let $\texttt{verb} = \{\texttt{jump}, \texttt{run}, ...\}$ denote the set of all verbs.
- $L(G_2)$ contains `Simon says` sentences, e.g.:
  `Simon says jump`, `Simon says run`, ...
- $|L(G_2)| = |\texttt{verb}|$
- Is the sentence `Simon says science` member of $L(G_2)$?

  I'm gonna have to science the shit out of this.
  Mark Watney in The Martian

➜ Allowing every word would still result in exceedingly unlikely sentences.

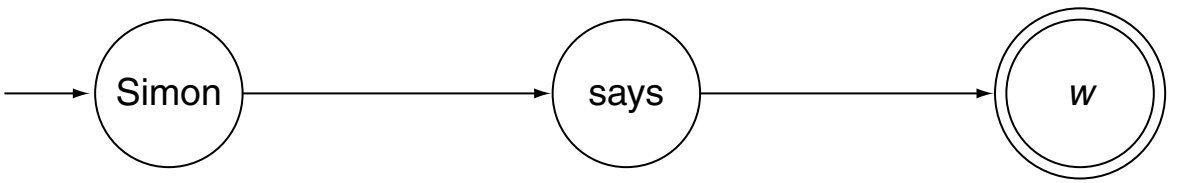
Remarks:

- Word classes may be either open or closed. An open class is one that commonly accepts the addition of new words, while a closed class is one to which new items are very rarely added. Open classes normally contain large numbers of words, while closed classes are much smaller. Typical open classes found in English and many other languages are nouns, verbs, adjectives, adverbs, and interjections. Typical closed classes are prepositions (or postpositions), determiners, conjunctions, and pronouns.
  The open-closed distinction is related to the distinction between lexical and functional categories, and to that between content words and function words. This is not universal: in many languages verbs and adjectives are closed classes, usually consisting of few members, and in Japanese the formation of new pronouns from existing nouns is relatively common.
  Words are added to open classes through such processes as compounding, derivation, coining, and borrowing. In English, for example, new nouns, verbs, etc. are being added to the language constantly (including by the common process of verbing and other types of conversion, where an existing word comes to be used in a different part of speech). [Wikipedia]

Language Models

Example: Statistical Language Model

Grammar G_3 as probabilistic automaton:



$w \in T$	$P(w)$
`jump`	0.05
`run`	0.03
⋮	⋮
`science`	0.002
⋮	⋮

where w is a random variable over a vocabulary T.

Generated language:

- $L(G_3)$ contains every three-word sentence starting with `Simon says` followed by a word w from T with probability $P(w) > \tau$ where τ is a threshold.
- Put another way, G_3 maps every sentence s that can be formed over its vocabulary Σ to a probability $P(s)$ so that

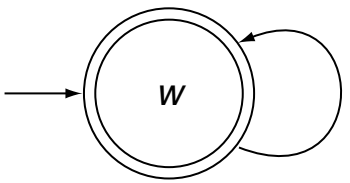
$$\sum_{s \in \Sigma^*} P(s) = 1$$

In general, probabilistic automata can be used to generate arbitrary documents.

Language Models

Example: Statistical Language Model

Grammar G_4 as probabilistic automaton:



$w \in T$	$P(w)$
$\perp$	0.2
`the`	0.2
`a`	0.1
`that`	0.04
`says`	0.03

$w \in T$	$P(w)$
`likes`	0.02
`Simon`	0.01
`Mark`	0.01
`science`	0.002
⋮	⋮

where w is a random variable over a vocabulary T.

Generated language:

- $\perp$ denotes the probability that the automaton stops.
- $L(G_4)$ contains all sentences that can be formed over the vocabulary T, assigning a membership probability to each one, e.g.:
  $s =$ `Simon says that Mark likes science` $\perp$
  $P(s) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2 = 0.0000000000048 = 4.8 \cdot 10^{-12}$
- Suppose every document were generated by its own language model $\mathbf{d}$.

➜ Given a query q, $P(\mathbf{d}_1 \mid q) > P(\mathbf{d}_2 \mid q)$ may indicate that d_1 is more relevant to q than d_2.

Remarks:

- This is the most basic form of a statistical language model, called a unigram (1-gram) language model. It assumes term independence and models language generation as a Markov process (0th order). Given sentence $s = t_1 \dots t_{|s|}$:

$$P(s) = P(t_1) \cdot \ldots \cdot P(t_{|s|})$$

- Dependent on the application, term dependence is modeled with conditional probabilities. Given sentence $s = t_1 \dots t_{|s|}$:

$$P(s) = P(t_1) \cdot P(t_2 \mid t_1) \cdot P(t_3 \mid t_1 t_2) \cdot \ldots \cdot P(t_{|s|} \mid t_1 \dots t_{|s|-1})$$

  The conditional probabilities need to be estimated from training data, and the longer the context of preceding words in the condition, the more data are needed. In practice, the probabilities are conditioned with up to n preceding words, obtaining an n-gram language model, where typically $n \in [1, 5]$.

- The stop probability $P(\perp)$ is required to fulfill the condition that all probabilities of the generated language sum to 1. In practice, however, it is usually omitted since doing so is rank-preserving when comparing probabilities among different language models.

- Since every ordering of the terms in a sentence has the same probability under the unigram model, one might also employ the multinomial distribution to compute $P(s)$ which would yield the probability of s regardless of the word order. Again, this is not necessary in practice, since differences in probability are constant and therefore rank-preserving when comparing probabilities among different language models.

Language Models

Retrieval Model $\mathcal{R} = \langle \mathbf{D}, \mathbf{Q}, \rho \rangle$ [Generic Model] [Boolean] [VSM] [BIM] [BM25] [LSI] [ESA] [LM]

Document representations $\mathbf{D}$.

- $T = \{t_1, \ldots, t_m\}$ is the set of m index terms (stemmed words).
- $p(t \mid d)$ is the probability of generating t given d.
- $\mathbf{d} = \{(t_1, p(t_1 \mid d), \ldots, (t_m, p(t_m \mid d))\}$ is a probability distribution over T.

Query representations $\mathbf{Q}$.

- $\mathbf{q} = (t_1, \ldots, t_{|q|})$, where $t_i \in T$, is a sequence of index terms.

Relevance function ρ.

- $\rho(d, q) = P(\mathbf{d} \mid \mathbf{q})$, the query likelihood model.
- R^+ is a set of documents relevant to q obtained via relevance feedback.
- $\mathbf{R}^+ = \{(t_1, p(t_1 \mid R^+), \ldots, (t_m, p(t_m \mid R^+))\}$ is a probability distribution over T.
- $\rho(d, q) = \varphi_{KL}(\mathbf{d}, \mathbf{R}^+)$, the relevance model.

Language Models

Retrieval Model $\mathcal{R} = \langle \mathbf{D}, \mathbf{Q}, \rho \rangle$ [Generic Model] [Boolean] [VSM] [BIM] [BM25] [LSI] [ESA] [LM]

Document representations $\mathbf{D}$.

- $T = \{t_1, \ldots, t_m\}$ is the set of m index terms (stemmed words).
- $p(t \mid d)$ is the probability of generating t given d.
- $\mathbf{d} = \{(t_1, p(t_1 \mid d), \ldots, (t_m, p(t_m \mid d))\}$ is a probability distribution over T.

Query representations $\mathbf{Q}$.

- $\mathbf{q} = (t_1, \ldots, t_{|q|})$, where $t_i \in T$, is a sequence of index terms.

Relevance function ρ.

- $\rho(d, q) = P(\mathbf{d} \mid \mathbf{q})$, the query likelihood model.
- R^+ is a set of documents relevant to q obtained via relevance feedback.
- $\mathbf{R}^+ = \{(t_1, p(t_1 \mid R^+), \ldots, (t_m, p(t_m \mid R^+))\}$ is a probability distribution over T.
- $\rho(d, q) = \varphi_{KL}(\mathbf{d}, \mathbf{R}^+)$, the relevance model.

Language Models

Relevance Function ρ: Derivation

Let $\mathbf{d}$ denote a language model for document d, and $\mathbf{q}$ the sequence of query terms from query q. Then the query likelihood model is derived as follows:

$$P(\mathbf{d} \mid \mathbf{q}) \quad = \quad \frac{P(\mathbf{q} \mid \mathbf{d}) \cdot P(\mathbf{d})}{P(\mathbf{q})} \tag{1}$$

(1) Application of Bayes' rule.

Language Models

Relevance Function ρ: Derivation

Let $\mathbf{d}$ denote a language model for document d, and $\mathbf{q}$ the sequence of query terms from query q. Then the query likelihood model is derived as follows:

$$P(\mathbf{d} \mid \mathbf{q}) = \frac{P(\mathbf{q} \mid \mathbf{d}) \cdot P(\mathbf{d})}{P(\mathbf{q})} \quad (1)$$

$$\overset{\text{rank}}{=} P(\mathbf{q} \mid \mathbf{d}) \cdot P(\mathbf{d}) \quad (2)$$

(1) Application of Bayes' rule.

(2) Rank-preserving omission of $P(\mathbf{q})$; it does not depend on $\mathbf{d}$.

Language Models

Relevance Function ρ: Derivation

Let $\mathbf{d}$ denote a language model for document d, and $\mathbf{q}$ the sequence of query terms from query q. Then the query likelihood model is derived as follows:

$$P(\mathbf{d} \mid \mathbf{q}) = \frac{P(\mathbf{q} \mid \mathbf{d}) \cdot P(\mathbf{d})}{P(\mathbf{q})} \quad (1)$$

$$\overset{\text{rank}}{=} P(\mathbf{q} \mid \mathbf{d}) \cdot P(\mathbf{d}) \quad (2)$$

$$= P(\mathbf{q} \mid \mathbf{d}) \quad (3)$$

(1) Application of Bayes' rule.

(2) Rank-preserving omission of $P(\mathbf{q})$; it does not depend on $\mathbf{d}$.

(3) Assume $P(\mathbf{d})$ is uniform for all $d \in D$, thereby canceling its influence.
This assumption is not required; as a prior, $P(\mathbf{d})$ can be used as a weight of relative importance of d (e.g., PageRank, quality, etc.).

Language Models

Relevance Function ρ: Derivation

Given a language model $\mathbf{d}$ of document d and a sequence $\mathbf{q}$ of the terms in query q, compute the probability that $\mathbf{q}$ has been generated by $\mathbf{d}$.

$$P(\mathbf{q} \mid \mathbf{d}) \quad = \quad P(t_1, \ \ldots, \ t_{|q|} \mid \mathbf{d}) \tag{4}$$

(4) Inflating $\mathbf{q}$.

Language Models

Relevance Function ρ: Derivation

Given a language model $\mathbf{d}$ of document d and a sequence $\mathbf{q}$ of the terms in query q, compute the probability that $\mathbf{q}$ has been generated by $\mathbf{d}$.

$$P(\mathbf{q} \mid \mathbf{d}) = P(t_1, \ldots, t_{|q|} \mid \mathbf{d}) \quad (4)$$

$$= \prod_{i=1}^{|q|} P(t_i \mid \mathbf{d}) \quad (5)$$

(4) Inflating $\mathbf{q}$.

(5) Assuming independence between terms.

Language Models

Relevance Function ρ: Derivation

Given a language model $\mathbf{d}$ of document d and a sequence $\mathbf{q}$ of the terms in query q, compute the probability that $\mathbf{q}$ has been generated by $\mathbf{d}$.

$$P(\mathbf{q} \mid \mathbf{d}) = P(t_1, \ldots, t_{|q|} \mid \mathbf{d}) \tag{4}$$

$$= \prod_{i=1}^{|q|} P(t_i \mid \mathbf{d}) \tag{5}$$

$$= \prod_{t \in \mathbf{q}} P(t \mid \mathbf{d})^{\mathit{tf}(t,q)} \tag{6}$$

(4) Inflating $\mathbf{q}$.

(5) Assuming independence between terms.

(6) Combine duplicate occurrences of term t in query q.

This corresponds to the multinomial distribution, albeit omitting its factor $|d| / \prod_{t \in q} \mathit{tf}(t, q)$, which counts the permutations of q's terms but is constant for q.

Language Models

Relevance Function ρ: Derivation

Given a language model $\mathbf{d}$ of document d and a sequence $\mathbf{q}$ of the terms in query q, compute the probability that $\mathbf{q}$ has been generated by $\mathbf{d}$.

$$P(\mathbf{q} \mid \mathbf{d}) = P(t_1, \ldots, t_{|q|} \mid \mathbf{d}) \tag{4}$$

$$= \prod_{i=1}^{|q|} P(t_i \mid \mathbf{d}) \tag{5}$$

$$= \prod_{t \in \mathbf{q}} P(t \mid \mathbf{d})^{\mathit{tf}(t,q)} \tag{6}$$

(4) Inflating $\mathbf{q}$.

(5) Assuming independence between terms.

(6) Combine duplicate occurrences of term t in query q.
This corresponds to the multinomial distribution, albeit omitting its factor $|d| / \prod_{t \in q} \mathit{tf}(t, q)$, which counts the permutations of q's terms but is constant for q.

Language Models

Relevance Function ρ: Derivation

Given a language model $\mathbf{d}$ of document d and a sequence $\mathbf{q}$ of the terms in query q, compute the probability that $\mathbf{q}$ has been generated by $\mathbf{d}$.

$$P(\mathbf{q} \mid \mathbf{d}) = P(t_1, \ldots, t_{|q|} \mid \mathbf{d}) \tag{4}$$

$$\overset{\text{rank}}{=} \sum_{i=1}^{|q|} \log P(t_i \mid \mathbf{d}) \tag{5}$$

$$= \prod_{t \in \mathbf{q}} P(t \mid \mathbf{d})^{\mathit{tf}(t,q)} \tag{6}$$

(4) Inflating $\mathbf{q}$.

(5) Assuming independence between terms.
Rank-preserving logarithmization to handle small probabilities.

(6) Combine duplicate occurrences of term t in query q.
This corresponds to the multinomial distribution, albeit omitting its factor $|d| / \prod_{t \in q} \mathit{tf}(t, q)$, which counts the permutations of q's terms but is constant for q.

Language Models

Relevance Function ρ: Estimation

Let t denote a term from the set of index terms T of document collection D. The construction of a language model $\mathbf{d}$ to represent document d is done as follows.

$$P(t \mid \mathbf{d}) = \frac{\mathit{tf}(t,d)}{|d|}, \qquad \text{where} \quad \sum_{t \in T} P(t \mid \mathbf{d}) = 1 \tag{7}$$

(7) Maximum likelihood estimation of t's probability under the assumed language model $\mathbf{d}$ for document d's topic, given the observed sample d.
Problem: $P(t \mid \mathbf{d}) = 0$ for $t \notin d$, causing $P(\mathbf{q} \mid \mathbf{d}) = 0$ if $t \in q$.

Language Models

Relevance Function ρ: Estimation

Let t denote a term from the set of index terms T of document collection D. The construction of a language model $\mathbf{d}$ to represent document d is done as follows.

$$P(t \mid \mathbf{d}) = \frac{\mathit{tf}(t,d)}{|d|}, \quad \text{where} \quad \sum_{t \in T} P(t \mid \mathbf{d}) = 1 \tag{7}$$

$$P(t \mid \mathbf{D}) = \frac{\sum_{d \in D} \mathit{tf}(t,d)}{\sum_{d \in D} |d|}, \quad \text{where} \quad \sum_{t \in T} P(t \mid \mathbf{D}) = 1 \tag{8}$$

(7) Maximum likelihood estimation of t's probability under the assumed language model $\mathbf{d}$ for document d's topic, given the observed sample d.
Problem: $P(t \mid \mathbf{d}) = 0$ for $t \notin d$, causing $P(\mathbf{q} \mid \mathbf{d}) = 0$ if $t \in q$.

(8) Maximum likelihood estimation of t's probability in a language model $\mathbf{D}$ for D.

Language Models

Relevance Function ρ: Estimation

Let t denote a term from the set of index terms T of document collection D. The construction of a language model $\mathbf{d}$ to represent document d is done as follows.

$$P(t \mid \mathbf{d}) = \frac{\mathit{tf}(t,d)}{|d|}, \quad \text{where} \quad \sum_{t \in T} P(t \mid \mathbf{d}) = 1 \tag{7}$$

$$P(t \mid \mathbf{D}) = \frac{\sum_{d \in D} \mathit{tf}(t,d)}{\sum_{d \in D} |d|}, \quad \text{where} \quad \sum_{t \in T} P(t \mid \mathbf{D}) = 1 \tag{8}$$

$$P(t \mid \mathbf{d})' = (1-\lambda) \cdot P(t \mid \mathbf{d}) \; + \; \lambda \cdot P(t \mid \mathbf{D}) \tag{9}$$

(7) Maximum likelihood estimation of t's probability under the assumed language model $\mathbf{d}$ for document d's topic, given the observed sample d.
Problem: $P(t \mid \mathbf{d}) = 0$ for $t \notin d$, causing $P(\mathbf{q} \mid \mathbf{d}) = 0$ if $t \in q$.

(8) Maximum likelihood estimation of t's probability in a language model $\mathbf{D}$ for D.

(9) Jelinek-Mercer smoothing: linear interpolation of language models $\mathbf{d}$ and $\mathbf{D}$.

Remarks:

- A document d must be considered just one sample of all possible documents that can be generated by the language model of d's topic. Therefore, it may be that it does not contain all terms that are potentially relevant.
- At the same time, a query q may contain terms that are not generally found in documents generated by the language models relating to q's relevant documents' topic. The appearance of such a misleading term should not penalize the relevance score of a potentially relevant document to zero when it does not contain the term.
- Moreover, the terms occurring just once in a document may be assigned an overestimated probability of being generated by the language model underlying a document's topic.
- Smoothing addresses all of these problems by shifting a portion of the probability mass from the terms occurring in a document to all other terms; just enough to give them a non-zero probability.
- Smoothing implements a major part of the term weighting component. Unsmoothed language models and language models with poorly optimized smoothing parameters perform significantly worse than language models with optimized smoothing.

Language Models

Relevance Function ρ: Estimation

Taking into account the length of a document d yields an alternative smoothing method.

$$P(t \mid \mathbf{d})' \quad = \; (1-\lambda) \cdot P(t \mid \mathbf{d}) \; + \; \lambda \cdot P(t \mid \mathbf{D}) \tag{9}$$

(9) Jelinek-Mercer smoothing: linear interpolation of language models $\mathbf{d}$ and $\mathbf{D}$.

Language Models

Relevance Function ρ: Estimation

Taking into account the length of a document d yields an alternative smoothing method.

$$P(t \mid \mathbf{d})' = (1-\lambda) \cdot P(t \mid \mathbf{d}) + \lambda \cdot P(t \mid \mathbf{D}) \tag{9}$$

$$\lambda = \frac{\alpha}{|d| + \alpha} \tag{10}$$

(9) Jelinek-Mercer smoothing: linear interpolation of language models $\mathbf{d}$ and $\mathbf{D}$.

(10) Dirichlet smoothing: adjust λ with respect to the length of document d. The longer a document d, the more trustworthy its language model $\mathbf{d}$ becomes.

Language Models

Relevance Function ρ: Estimation

Taking into account the length of a document d yields an alternative smoothing method.

$$P(t \mid \mathbf{d})' = (1-\lambda) \cdot P(t \mid \mathbf{d}) + \lambda \cdot P(t \mid \mathbf{D}) \tag{9}$$

$$\lambda = \frac{\alpha}{|d| + \alpha} \tag{10}$$

$$P(t \mid \mathbf{d})'' = \frac{\mathit{tf}(t,d) + \alpha \cdot P(t \mid \mathbf{D})}{|d| + \alpha} \tag{11}$$

(9) Jelinek-Mercer smoothing: linear interpolation of language models $\mathbf{d}$ and $\mathbf{D}$.

(10) Dirichlet smoothing: adjust λ with respect to the length of document d. The longer a document d, the more trustworthy its language model $\mathbf{d}$ becomes.

(11) Substitution of λ in $P(t \mid \mathbf{d})'$.

Remarks:

- From a Bayesian point of view, this corresponds to the expected value of the posterior distribution, using a symmetric Dirichlet distribution with parameter α as a prior. [Wikipedia]
- Finding a value for λ that works well in all cases is difficult. A small λ results in less smoothing and yields a retrieval semantic closer to Boolean AND, whereas a high λ is closer to Boolean OR and rather ranks documents by the number of matching query terms. The choice of λ very much depends on what kinds of queries are expected.
- Successful values for short and long queries are $\lambda = 0.1$ and $\lambda = 0.7$, respectively. Furthermore, typically $1000 < \alpha < 2000$. [plot] [Croft 2015]
- Dirichlet smoothing performs better for keyword queries, Jelinek-Mercer smoothing performs better for verbose queries. [Schütze 2014]
- In practice, language models are sensitive to smoothing parameters. They should always be tuned to the retrieval scenario at hand.

Language Models

Relevance Function ρ: Example

Let $q = \texttt{president lincoln}$ and let $d_1 \in D$ be a document from a collection D.

Assumptions:

- $\mathit{tf}(\texttt{president}, d_1) = 15$ and $\sum_{d \in D} \mathit{tf}(\texttt{president}, d) = 160,000$
- $\mathit{tf}(\texttt{lincoln}, d_1) = 25$ and $\sum_{d \in D} \mathit{tf}(\texttt{lincoln}, d) = 2,400$
- $|d_1| = 1,800$ and $|D| = 500,000$ at $|d|_{\text{avg}} = 2,000$, yielding 10^9 terms.
- $\alpha = |d|_{\text{avg}} = 2,000$

$$
\begin{aligned}
\rho(\mathbf{d}_1, \mathbf{q}) &= \log \frac{15 + 2000 \cdot (1.6 \cdot 10^5 / 10^9)}{1800 + 2000} &+ \log \frac{25 + 2000 \cdot (2400/10^9)}{1800 + 2000} \\
&= \log(15.32/3800) &+ \log(25.005/3800) \\
&= -5.51 &+ -5.02 \\
&= -10.53
\end{aligned}
$$

Logarithmization yields negative relevance scores; recall that only the ranking among documents is important, not the scores themselves.

Language Models

Relevance Function ρ: Example

Let $q = \texttt{president lincoln}$ and let $d_1 \in D$ be a document from a collection D.

Assumptions:

- $tf(\texttt{president}, d_1) = 15$ and $\sum_{d \in D} tf(\texttt{president}, d) = 160,000$
- $tf(\texttt{lincoln}, d_1) = 25$ and $\sum_{d \in D} tf(\texttt{lincoln}, d) = 2,400$
- $|d_1| = 1,800$ and $|D| = 500,000$ at $|d|_{\text{avg}} = 2,000$, yielding 10^9 terms.
- $\alpha = |d|_{\text{avg}} = 2,000$

D	president	lincoln	LM	#	BM25	#
d_1	15	25	-10.53	1	20.66	1
d_2	15	1	-13.75	3	12.74	4
d_3	15	0	-19.05	5	5.00	5
d_4	1	25	-12.99	2	18.20	2
d_5	0	25	-14.40	4	15.66	3

Language Models

Relevance Function ρ: Summary

$$\rho(\mathbf{d}, \mathbf{q}) \quad = \quad P(\mathbf{d} \mid \mathbf{q}) \quad \propto \quad P(\mathbf{d}) \cdot \prod_{i=1}^{|q|} \frac{\mathit{tf}(t_i, d) + \alpha \cdot \frac{\sum_{d \in D} \mathit{tf}(t_i, d)}{\sum_{d \in D} |d|}}{|d| + \alpha}$$

Assumptions:

1. The user has a mental model of the desired document and generates the query from that model.
2. The equation represents a probability estimate that the document the user had in mind was in fact this one.
3. Independence of word occurrence in documents.
4. Terms not in query q are equally likely to occur in relevant and irrelevant documents.
5. The prior $P(\mathbf{d})$ may be chosen uniform for all documents, or to boost more important documents.

Language Models

Query Refinement: Relevance Feedback

Given a query q, let R^* denote the subset of relevant documents from document collection D. Every $d \in R^*$ and q are samples drawn from their relevance model $\mathbf{R}^*$.

$$P(\mathbf{d} \mid \mathbf{q}) \quad \overset{\text{rank}}{=} \quad -\mathit{KL}(\mathbf{R}^* \mid\mid \mathbf{d}) \tag{1}$$

(1) Rank-preserving approximation by measuring the negative statistical difference between the language model $\mathbf{d}$ and that of the set $\mathbf{R}^*$ of relevant documents to query q using the Kullback–Leibler (KL) divergence measure.

Language Models

Query Refinement: Relevance Feedback

Given a query q, let R^* denote the subset of relevant documents from document collection D. Every $d \in R^*$ and q are samples drawn from their relevance model $\mathbf{R}^*$.

$$P(\mathbf{d} \mid \mathbf{q}) \overset{\text{rank}}{=} -\sum_{t \in T} P(t \mid \mathbf{R}^*) \log \frac{P(t \mid \mathbf{R}^*)}{P(t \mid \mathbf{d})} \tag{1}$$

(1) Rank-preserving approximation by measuring the negative statistical difference between the language model $\mathbf{d}$ and that of the set $\mathbf{R}^*$ of relevant documents to query q using the Kullback–Leibler (KL) divergence measure.

Language Models

Query Refinement: Relevance Feedback

Given a query q, let R^* denote the subset of relevant documents from document collection D. Every $d \in R^*$ and q are samples drawn from their relevance model $\mathbf{R}^*$.

$$P(\mathbf{d} \mid \mathbf{q}) \overset{\text{rank}}{=} -\sum_{t \in T} P(t \mid \mathbf{R}^*) \log \frac{P(t \mid \mathbf{R}^*)}{P(t \mid \mathbf{d})} \tag{1}$$

$$= \sum_{t \in T} P(t \mid \mathbf{R}^*) \log P(t \mid \mathbf{d}) - \sum_{t \in T} P(t \mid \mathbf{R}^*) \log P(t \mid \mathbf{R}^*) \tag{2}$$

(1) Rank-preserving approximation by measuring the negative statistical difference between the language model $\mathbf{d}$ and that of the set $\mathbf{R}^*$ of relevant documents to query q using the Kullback–Leibler (KL) divergence measure.

(2) Rearrangement.

Language Models

Query Refinement: Relevance Feedback

Given a query q, let R^* denote the subset of relevant documents from document collection D. Every $d \in R^*$ and q are samples drawn from their relevance model $\mathbf{R}^*$.

$$P(\mathbf{d} \mid \mathbf{q}) \overset{\text{rank}}{=} -\sum_{t \in T} P(t \mid \mathbf{R}^*) \log \frac{P(t \mid \mathbf{R}^*)}{P(t \mid \mathbf{d})} \tag{1}$$

$$= \sum_{t \in T} P(t \mid \mathbf{R}^*) \log P(t \mid \mathbf{d}) - \sum_{t \in T} P(t \mid \mathbf{R}^*) \log P(t \mid \mathbf{R}^*) \tag{2}$$

$$\overset{\text{rank}}{=} \sum_{t \in T} P(t \mid \mathbf{R}^*) \log P(t \mid \mathbf{d}) \tag{3}$$

(1) Rank-preserving approximation by measuring the negative statistical difference between the language model $\mathbf{d}$ and that of the set $\mathbf{R}^*$ of relevant documents to query q using the Kullback–Leibler (KL) divergence measure.

(2) Rearrangement.

(3) Rank-preserving omission of the second sum; it does not depend on $\mathbf{d}$.

Remarks:

- The Kullback-Leibler divergence $KL(P \mid\mid Q)$ (also called relative entropy) is a measure of how probability distribution Q diverges from an expected probability distribution P. It is a distribution-wise asymmetric measure. A Kullback-Leibler divergence of 0 indicates that we can expect similar, if not the same, behavior of two different distributions, while a Kullback-Leibler divergence of 1 indicates that the two distributions behave in such a different manner that the expectation given the first distribution approaches zero.
  In applications, P typically represents the "true" distribution of data, observations, or a precisely calculated theoretical distribution, while Q typically represents a theory, model, description, or approximation of P. [Wikipedia]
- Idea: Estimate $P(d \mid \mathbf{R}^*)$ directly, i.e., the probability that the relevance model generates document d. This is called the document likelihood model, but it does not work in practice: the estimates for d heavily depend on its length $|d|$ and are therefore hardly comparable for documents of different lengths.
- Idea: Simply estimating $P(t \mid \mathbf{R}^*)$ with the maximum likelihood estimate of t occurring in q:

  $$P(t \mid \mathbf{R}^*) = \frac{\mathit{tf}(t,q)}{|q|}$$

  yields

  $$P(\mathbf{d} \mid \mathbf{q}) \overset{\text{rank}}{=} \sum_{t \in T} \frac{\mathit{tf}(t,q)}{|q|} \log P(t \mid \mathbf{d}) = \frac{1}{|q|} \sum_{t \in T} \log P(t \mid \mathbf{d})^{\mathit{tf}(t,q)}$$

  which is equivalent to the query likelihood model.

Language Models

Query Refinement: Relevance Feedback

Since R^* is unknown at query time, we cannot approximate its language model directly. But we can exploit that, by definition, q has been sampled from $\mathbf{R}^*$.

$$P(t \mid \mathbf{R}^*) \quad \approx \quad P(t \mid \mathbf{q}) \quad = \quad P(t \mid t_1, \ \ldots, \ t_{|q|}) \quad \text{for } t_i \in \mathbf{q} \qquad (4)$$

(4) Approximation as probability of observing term t given query q: sampling the sequence $\mathbf{q} = (t_1, \ \ldots, \ t_{|q|})$ from $\mathbf{R}^*$, what is the probability of sampling t next?

Language Models

Query Refinement: Relevance Feedback

Since R^* is unknown at query time, we cannot approximate its language model directly. But we can exploit that, by definition, q has been sampled from $\mathbf{R}^*$.

$$P(t \mid \mathbf{R}^*) \approx P(t \mid \mathbf{q}) = P(t \mid t_1, \ldots, t_{|q|}) \quad \text{for } t_i \in \mathbf{q} \tag{4}$$

$$= \frac{P(t, t_1, \ldots, t_{|q|})}{P(t_1, \ldots, t_{|q|})} \tag{5}$$

(4) Approximation as probability of observing term t given query q: sampling the sequence $\mathbf{q} = (t_1, \ldots, t_{|q|})$ from $\mathbf{R}^*$, what is the probability of sampling t next?

(5) Definition of conditional probability.

Language Models

Query Refinement: Relevance Feedback

Since R^* is unknown at query time, we cannot approximate its language model directly. But we can exploit that, by definition, q has been sampled from $\mathbf{R}^*$.

$$P(t \mid \mathbf{R}^*) \approx P(t \mid \mathbf{q}) = P(t \mid t_1, \ldots, t_{|q|}) \quad \text{for } t_i \in \mathbf{q} \tag{4}$$

$$= \frac{P(t, t_1, \ldots, t_{|q|})}{P(t_1, \ldots, t_{|q|})} \tag{5}$$

$$= \frac{P(t, t_1, \ldots, t_{|q|})}{\sum_{t \in T} P(t, t_1, \ldots, t_{|q|})} \tag{6}$$

(4) Approximation as probability of observing term t given query q: sampling the sequence $\mathbf{q} = (t_1, \ldots, t_{|q|})$ from $\mathbf{R}^*$, what is the probability of sampling t next?

(5) Definition of conditional probability.

(6) Ensures additivity of the model.

Language Models

Query Refinement: Relevance Feedback

Let $R^+ \subseteq R^*$ be a set of documents relevant to query q, which have been obtained via relevance feedback.

$$P(t,\ t_1,\ \ldots,\ t_{|q|}) \quad \approx \sum_{d \in R^+} P(\mathbf{d}) \cdot P(t,\ t_1,\ \ldots,\ t_{|q|} \mid \mathbf{d}) \qquad (7)$$

(7) Approximation based on the law of total probability, using the language models of the individual relevant documents.

Language Models

Query Refinement: Relevance Feedback

Let $R^+ \subseteq R^*$ be a set of documents relevant to query q, which have been obtained via relevance feedback.

$$P(t,\ t_1,\ \ldots,\ t_{|q|}) \approx \sum_{d \in R^+} P(\mathbf{d}) \cdot P(t,\ t_1,\ \ldots,\ t_{|q|} \mid \mathbf{d}) \tag{7}$$

$$= \sum_{d \in R^+} P(\mathbf{d}) \cdot P(t \mid \mathbf{d}) \cdot \prod_{i=1}^{|q|} P(t_i \mid \mathbf{d}) \tag{8}$$

(7) Approximation based on the law of total probability, using the language models of the individual relevant documents.

(8) Assumption that the query's terms $t_1,\ \ldots,\ t_{|q|}$ are independent of one another, as well as from t.

Language Models

Query Refinement: Relevance Feedback

$$\rho(\mathbf{d}, \mathbf{q}) \;=\; P(\mathbf{d} \mid \mathbf{q}) \;\propto\; \sum_{t \in T} \frac{\sum_{d' \in R^+} P(\mathbf{d}') \cdot P(t \mid \mathbf{d}') \cdot \prod_{i=1}^{|q|} P(t_i \mid \mathbf{d}')}{\sum_{t \in T} \sum_{d' \in R^+} P(\mathbf{d}') \cdot P(t \mid \mathbf{d}') \cdot \prod_{i=1}^{|q|} P(t_i \mid \mathbf{d}')} \cdot \log P(t \mid \mathbf{d})$$

Retrieval:

1. Given query q, rank the documents in D by their query likelihood score.
2. Use the top-ranked 10–50 documents as pseudo-relevance feedback R^+.
3. Compute the relevance model probabilities.
4. Rank documents by their KL divergence score as computed above.

Language Models

Discussion

Advantages:

- Mathematically precise, conceptually simple, computationally tractable, and intuitively appealing
- Competitive retrieval performance

Disadvantages:

- Requires extensive tuning
- Assumption of equivalence between document and information need representation is unrealistic
- Difficult to represent the fact that a query is just one of many possible queries to describe a particular need