6. Language Modeling for Retrieval

Prof. Dr. Goran Glavaš

Center for AI and Data Science (CAIDAS)
Fakultät für Mathematik und Informatik
Universität Würzburg

After this lecture, you'll...

- Know what a language model is
- Understand differences between different language models (unigram, bigram)
- Understand how to use language modeling for information retrieval
- Learn about different smoothing schemes for LM for IR
- Be able to compare LM for IR with vector space model and classic probabilistic models

Outline

- Recap of Lecture #5
- Language Models
  - Unigram LM
  - Bigram LM
- Query likelihood model for ranking
- Smoothing schemes
- Projects
  - Topics explained
  - Organization

Recap of the previous lecture

- Probabilistic retrieval
  - **Q:** Why probability theory in IR, and why probabilistic ranking?
  - **Q:** What are the uncertainties of the IR process that we model probabilistically?
- Probability ranking principle
  - **Q:** What does Robertson's probabilistic ranking principle say?
  - **Q:** How do we formalize the probability ranking principle?
- Probabilistic ranking
  - **Q:** What is the ranking task formulation in the probabilistic setting?
  - **Q:** Starting from (log-)odds of relevance, how do we derive the general probabilistic ranking score?
- Binary independence model and extensions
  - **Q:** What assumptions does binary independence model introduce?
  - **Q:** What does the ranking function look like under these assumptions?
  - **Q:** Derive the BIM ranking function with and without relevance judgements
  - **Q:** How do Two Poisson, BM11, and BM25 extend BIM? What assumptions do they introduce?

Recap of the previous lecture

- The ranking score at the core of all probabilistic models:

$$log\left(\frac{P(D|Q,r)}{P(D|Q,\bar{r})}\right)$$

- Ranking function of Binary Independence Model
  - Without (left) and with (right) relevance judgements

$$rel(D,Q) = \sum_{t\in Q} log\left(\frac{P(D_t|Q,r)}{P(D_t|Q,\bar{r})}\right)$$
$$= \sum_{t\in Q} log\left(\frac{0.5}{\frac{N_t}{N}}\right)$$
$$= \sum_{t\in Q} log\left(0.5 \cdot \frac{N}{N_t}\right)$$

$$rel(D,Q) = \sum_{t\in Q} log\left(\frac{P(D_t|Q,r)}{P(D_t|Q,\bar{r})}\right)$$
$$= \sum_{t\in Q} log\left(\frac{\frac{r_t+0.5}{R+1}}{\frac{N_t-r_t+0.5}{N-R+1}}\right)$$
$$= \sum_{t\in Q} log\left(\frac{(r_t+0.5)\cdot(N-R+1)}{(R+1)\cdot(N_t-r_t+0.5)}\right)$$

Binary independence model – example #1

- Example for BIM (without relevance judgements)
- Document collection consists of the following documents:
  - d_1: „Frodo and Sam stabbed orcs"
  - d_2: „Sam chased the orc with the sword"
  - d_3: „Sam took the sword"
- The query is: „Sam stabbed orc"

	d_1			d_2		d_3
t	Sam	stabbed	orcs	Sam	orc	Sam
$P(D_t \mid q, r)$	0.5	0.5	0.5	0.5	0.5	0.5
$P(D_t \mid q, \bar{r})$	3/3	1/3	2/3	3/3	2/3	3/3
w_t	0.5	1.5	0.75	0.5	0.75	0.5
$\sum w_t$	2.75			1.25		0.5

- Note: computations in this example are done without taking the logarithm

Binary independence model – example #2

- Example for BIM (with available relevance judgements)
- Document collection contains N = 30 documents, including:
  - d_1: „Frodo and Sam stabbed orcs"
  - d_2: „Sam chased the orc with the sword"
  - d_3: „Sam took the sword"
- The query is: „Sam stabbed orc"
- User has indicated R = 6 relevant documents for this query
- Query terms: t_1 = „Sam", t_2 = „stab", t_3 = „orc"
- Document frequencies of query terms in relevant documents and overall collection are given as follows:
  - $r_{t1} = 3$, $N_{t1} = 15$
  - $r_{t2} = 4$, $N_{t2} = 16$
  - $r_{t3} = 2$, $N_{t3} = 14$

Binary independence model – example #2

- Example for BIM (with available relevance judgements)

	d_1			d_2		d_3
t	Sam	stabbed	orcs	Sam	orc	Sam
$P(D\|Q,r) = \frac{r_t+0.5}{R+1}$	0.5	0.64	0.36	0.5	0.36	0.5
$P(D\|Q,\bar{r}) = \frac{N_t-r_t+0.5}{N-R+1}$	0.5	0.5	0.5	0.5	0.5	0.5
w_t	1	1.28	0.72	1	0.72	1
$\sum_t w_t$	3			1.72		1

- Note: computations in this example are done without taking the logarithm

Outline

- Recap of Lecture #5
- Language Models
  - Unigram LM
  - Bigram LM
- Query likelihood model for ranking
- Smoothing schemes

Language Modeling (for Information Retrieval)

- **Language models** are probabilistic models that capture the probabilities of sequences of words in a language
  - **Unigram model**: how likely is the word „frodo” to appear (in a language)?
    - P(„frodo”) = ?
  - **Bigram model**: given that current word is „frodo”, what is the probability of next word being „baggins”?
    - P(„baggins” | „frodo”) = ?
  - **Trigram model**: given the current sequence „frodo baggins”, what is the probability of next word being „shire”?
    - P(„shire” | „frodo baggins”) = ?

- **Q:** How do we estimate probabilities of words and sequences in a language?
  - I.e., What do we use as a representation of the language?

Language Modeling (for Information Retrieval)

- We use the instantiations of the language to estimate the probabilities of words and sequences
  - I.e., **large corpora** – the larger the corpora, it is the better approximation of the true word distributions in language

- In other applications we build language models largest corpora we can compile

- In information retrieval, we build language models
  1. From individual documents
  2. From the whole document collections

Language Modeling (for Information Retrieval)

- Language models for IR are also **probabilistic models**
- Language models for IR model the **query generation process**
- Given a documet d and a query q, what is the probability of query being sampled from the language model of the document
- In other words, we want to estimate $P(Q = q \mid D = d)$
  - **Q:** Compare this with the probability we estimated in classic probabilistic retrieval
    - $P(R = 1 \mid Q = q, D = d)$
- The probability of a document generating a query is directly the function according to which we rank the documents
  - I.e., We rank the documents in decreasing order of $P(Q = q \mid D = d)$
- **Key question:** how do we estimate $P(Q = q \mid D = d)$?

Language Bowl Metaphor

- Assume we have a document in with following occurrences of terms:
  - „frodo” (5x), „baggins” (3x), „sam” (3x), „shire” (2x), „gandalf” (2x), „orc” (1x)
- Let’s represent each term with balls of one color:
  - „frodo” -> 5 blue balls, „baggins” -> 3 red balls, „sam” -> 3 yellow balls
  - „shire” -> 2 green balls, „gandalf” -> 2 orange balls, „orc” -> 1 purple ball
- We put all balls into one bowl and randomly take them out one by one
- **Q:** What is the probability of drawing a yellow ball?
  - P(●) = P(„sam”) = 3 / (5 + 3 + 3 + 2 + 2 + 1) = 3 / 16
- **Q:** What is the probability of drawing first orange then blue ball?
  - Replacement: P(●, ●) = P(„gandalf”, „frodo”) = P(„gandalf”) * P(„frodo”) = 2/16 * 5/16
  - No replacement: P(●, ●) = P(„gandalf”, „frodo”) = P(„gandalf”) * P(„frodo” | „gandalf”)
    - = 2/16 * 5/15

Language Model – Generative Story

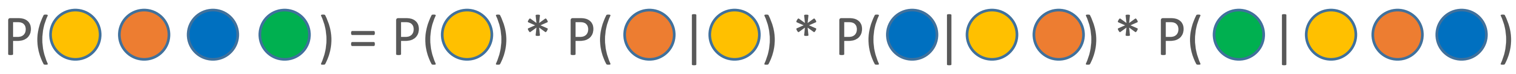
- Language model can be observed as a statistical model for generating data
- Example (toy language, consisting of four words):
  - P(„frodo”) = 0.3, P(„sam”) = 0.25, P(„gandalf”) = 0.35, P(„shire”) = 0.1
  - P(„sam” | „frodo”) = 0.4, P(„gandalf” | „frodo”) = 0.4, P(„shire” | „frodo”) = 0.2
- Generative process:
  1. Randomly draw the first word (e.g., from a uniform distribution)

  2. Draw the second word from conditional distribution of the first word (e.g., „frodo”)

- **Q:** What is the probability of the sequence „frodo shire”?

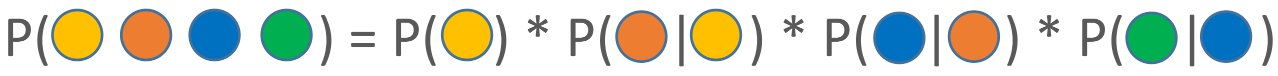
Types of Language Models

- We want to estimate the probability of the sequence:



- **Unigram language model**
  - Word independence = probability of the word does not depend on previous words
  - We ignore conditioning

- **Bigram language model**
  - The probability of word appearing depends only on the immediately preceding word
  - Conditioning only one one word before



- **Q:** N-gram models for $N \geq 3$ are rarely used in practice. Why?

Sparseness issue of language models

- Language models have a major issue
  - The longer the phrase, the harder it is to estimate its true probability in language
  - E.g., P(„bilbo” | „frodo ran around house found ring”) = ?

- Long phrases have very few appearances even in very large corpora
  - Impossible to compute reliable estimates of their conditional probabilities
  - This is why language models for N ≥ 3 are almost never used

- In practice, we use unigram and bigram language models
  - In IR setting, we build language models from invidual documents
    - Even bigram probability hard to estimate
  - In IR, we most often employ the unigram language model

Estimating probabilities

- For the unigram language model we need to estimate
  - P(term) for every term in the text
- For the bigram language model we additionally need to estimate
  - P(term | previous term) for every pair of terms that appear one after another
- **Q:** How do we estimate these?
  - Unigram language model
    - $P(t_i) = n_i / n_T$
    - n_i is the number of occurences of term t_i in the collection
    - n_T is the total number of word occurences (i.e., tokens) in the collection
  - Bigram language model
    - $P(t_i \mid t_{i-1}) = n(t_{i-1}, t_i) / n(t_{i-1})$
    - $n(t_{i-1}, t_i)$ is the number of occurences of bigram $t_{i-1}t_i$ in the collection
    - $n(t_{i-1})$ is the number of occurrences of term t_{i-1} in the collection

Estimating probabilities – example

- We are given a toy collection consisting of three documents
  - d_1: „Frodo and Sam stabbed orcs"
  - d_2: „Sam chased the orc with the sword"
  - d_3: „Sam took the sword"

- Estimating word probabilities for the unigram model:

t_i	Frodo	Sam	orc	chased	sword	...
$P(t_i)$	1/16	3/16	2/16	1/16	2/16	...

- Estimating the conditional probabilities for the bigram model:

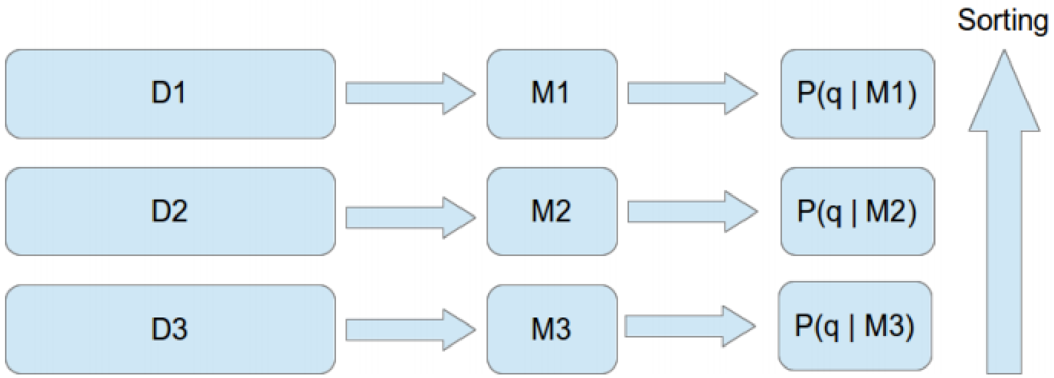
t_{i-1}, t_i	Frodo, chased	the, sword	the, orc	...
$P(t_i \| t_{i-1})$	0	2/3	1/3	...

Outline

- Recap of Lecture #5
- Language Models
  - Unigram LM
  - Bigram LM
- Query likelihood model for ranking
- Smoothing schemes

Query likelihood model for ranking

- Given a document collection D and a query q we need to estimate the probability P(q | d) for every document d in D
- In the **query likelihood model**, we estimate the probability P(q | d) as the probability that the language model built from d generates the query q
- Algorithm
  1. Compute the language model M_i for every document d_i in D
  2. Compute the probability P(q | Mi) for every language model M_i



- **Intuition**: Language models of relevant documents should assign higher probability for the query

Query likelihood model for ranking – example

- We are given a toy collection consisting of three documents
  - d_1: „Sam chased the orc with the sword"
  - d_2: „Frodo and Sam stabbed orcs"
  - d_3: „Sam took the sword"
- We are given the query „Sam and orc and sword"
- Let's rank the documents according to unigram LM for IR (ignore stopwords)

- **Step 1:** Compute language models of individual documents
  - M_1: P(„sam") = 0.25, P(„chase") = 0.25, P(„orc") = 0.25, P(„sword") = 0.25
  - M_2: P(„frodo") = 0.25, P(„sam") = 0.25, P(„stab") = 0.25, P(„orc") = 0.25
  - M_3: P(„sam") = 0.33, P(„took") = 0.33, P(„sword") = 0.33

Query likelihood model for ranking – example

- We are given a toy collection consisting of three documents
  - d_1: „Sam chased the orc with the sword"
  - d_2: „Frodo and Sam stabbed orcs"
  - d_3: „Sam took the sword"
- We are given the query „Sam and orc and sword"
- Let's rank the documents according to unigram LM for IR (ignore stopwords)

- **Step 2:** Let's compute the probabilities $P(q \mid M_i)$
  - $P(q \mid M_1) = P(\text{„sam"} \mid M_1) * P(\text{„orc"} \mid M_1) * P(\text{„sword"} \mid M_1) = 0.25 * 0.25 * 0.25$
  - $P(q \mid M_2) = P(\text{„sam"} \mid M_2) * P(\text{„orc"} \mid M_2) * P(\text{„sword"} \mid M_2) = 0.25 * 0.25 * \mathbf{0}$
  - $P(q \mid M_3) = P(\text{„sam"} \mid M_3) * P(\text{„orc"} \mid M_3) * P(\text{„sword"} \mid M_3) = 0.33 * \mathbf{0} * 0.33$
- **Q:** Is there any problem with query likelihoods given LMs of d_2 and d_3?

Outline

- Recap of Lecture #5
- Language Models
  - Unigram LM
  - Bigram LM
- Query likelihood model for ranking
- Smoothing schemes

Smoothing language models

- **Zero frequency problem:** Models we've considered so far give probability of **0** to queries containing any term that does not occur in the document
- We can prevent this by using **smoothing techniques**
- Smoothing techniques
  - Change the probability distribution of terms in the language model
  - Assign some small probability to unseen words
- Three prominent smoothing schemes
  - Laplace smoothing
  - Jelinek-Mercer smoothing
  - Dirichlet smoothing

Laplace smoothing

- **Laplace smoothing**
  1. Adds a fixed small count (often it's 1) to all word counts
  2. Renormalizes to get a probability distribution

$$P(t_i|M_d) = \frac{n_{i,d} + \alpha}{n_d + |V| \cdot \alpha}$$

  - The probability of any unseen word equals

$$P(t_{uns}|M_d) = \frac{\alpha}{n_d + |V| \cdot \alpha}$$

- **Q:** What might be a potential shortcoming of the Laplace smoothing?

Jelinek-Mercer smoothing

- Laplace smoothing assumes that all unseen words are equally likely
- **Jelinek-Mercer smoothing** (also known as **interpolated smoothing**)
  1. Additionally builds a language model M_D from the whole document collection D
  2. Interpolates between probabilities of the query according to the
     - **Local LM** – language model M_d built from the particular document d
     - **Global LM** – language model M_D built from the whole collection

$$P(t_i|M_d) = \lambda \cdot P(t_i|M_d) + (1-\lambda) \cdot P(t_i|M_D)$$

- The probability of a word unseen in the document d still gets some probability from the global language model
  - Probability of an unseen word depends on its frequency in whole collection
- **Q:** What if $P(t_i|M_D) = 0$?

Dirichlet smoothing

- **Dirichlet smoothing** can be seen as a generalization of the Laplace smoothing
  - Each word unseen in the document gets an artificial extra count
  - But the extra count is not fixed, depends on the global probability of the term
    - In this respect, Dirichlet smoothing is similar to Jelinek-Mercer smoothing

$$P(t_i|M_d) = \frac{n_{i,d} + \mu \cdot P(t_i|M_D)}{n_d + \mu}$$

- Less frequent words in the document get more probability from the global component
  - The value of the constant μ determines the scale of the global probability's contribution

Language models for IR vs. VSM

- Let's compare the query likelihood model with the VSM model

1. Do we have a term frequency component in LM?
   - **Q:** do query terms that are more frequent in the document contribute more to the relevance score?
   - **A:** Yes! $P(t_i) = \mathbf{n_i} / n_T$
2. Do we have a document frequency component in LM?
   - **Q:** does the global document frequency of the query term affect the relevance scores?
   - **A:** No! If we use Jelinek-Mercer or Dirichlet smoothing, we take into consideration collection frequency, but not document frequency
   - However, mixing term frequency (within document) and collection frequency has an effect similar to using IDF

Language models for IR vs. VSM

- Let's compare the query likelihood model with the VSM model

3. Does LM for IR account for different lengths of documents?
   - **Q:** Does it somehow normalize the frequencies of query terms in documents with the document length?
   - **A:** Yes! $P(t_i) = n_i / \mathbf{n_T}$

- LM for IR vs. VSM: **commonalities**
  1. Term frequency directly in the model
  2. Contributions of terms are normalized to account for document length
- LM for IR vs. VSM: **differences**
  1. LM for IR is based in probability theory, VSM in vector algebra
  2. Collection frequency (LM) vs. Document frequency (VSM)

Now you...

- Know what a language model is
- Understand differences between different language models (unigram, bigram)
- Understand how to use language modeling for information retrieval
- Are familiar with different smoothing schemes for LM for IR
- Are able to compare LM for IR with vector space model and classic probabilistic models