Build and querying document models

300958 Social Web Analytics

Week 4 Supplementary Material

Document Set

The set of four documents

- d_1: One one was a race horse
- d_2: Two two was one too
- d_3: One one won one race
- d_4: Two two won one too

We want to find the document that best matches the query "one won". To do this we convert the document text into document models and compare them to the query.

Document models

There are two document models that we can choose from: the vector space model (using TF-IDF weights and Cosine similarity), or the Language model (using Dirichlet smoothed weights and likelihood).

Vector Space Model

Building the document models

To create the document models, we first obtain the frequency $f_{d,t}$ of each word t in each document d.

	one	was	a	race	horse	two	too	won
d_1	2	1	1	1	1	0	0	0
d_2	1	1	0	0	0	2	1	0
d_3	3	0	0	1	0	0	0	1
d_4	1	0	0	0	0	2	1	1

We can then compute the IDF weight of each word ($\log(N/f_t)$, where N is the number of documents and f_t is the number of documents term t appears in). $N = 4$.

Note that we are using log base e (shown as ln on most calculators, and log in R), not log base 10 (shown as log on most calculators and log10 in R).

	one	was	a	race	horse	two	too	won
f_t	4	2	1	2	1	2	2	2

Giving the IDF weights for each word:

	one	was	a	race	horse	two	too	won
IDF	0	0.69	1.39	0.69	1.39	0.69	0.69	0.69

The TF weight is for each word in each document and given as $\log(f_{d,t} + 1)$.

	one	was	a	race	horse	two	too	won
d_1	1.10	0.69	0.69	0.69	0.69	0	0	0
d_2	0.69	0.69	0	0	0	1.10	0.69	0
d_3	1.39	0	0	0.69	0	0	0	0.69
d_4	0.69	0	0	0	0	1.10	0.69	0.69

So the final weights $w_{d,t}$ for each word in each document is given as TF×IDF.

	one	was	a	race	horse	two	too	won
d_1	0	0.48	0.96	0.48	0.96	0	0	0
d_2	0	0.48	0	0	0	0.76	0.48	0
d_3	0	0	0	0.48	0	0	0	0.48
d_4	0	0	0	0	0	0.76	0.48	0.48

Note that any words that have term frequency $f_{d,t} = 0$ obtain a weight of 0.

Building the query model

The query "one won" is a sequence of terms, just as in a document, so we can treat it as a document and use TF-IDF weights, where the IDF weights from from those computed above.

The term frequency of the query terms is:

	one	was	a	race	horse	two	too	won
q	1	0	0	0	0	0	0	1

The TF weight is:

	one	was	a	race	horse	two	too	won
q	0.69	0	0	0	0	0	0	0.69

Giving us the TF×IDF weights:

	one	was	a	race	horse	two	too	won
q	0	0	0	0	0	0	0	0.48

Computing the similarity of each document to the query

We now have a model of each document and the query.

	one	was	a	race	horse	two	too	won
d_1	0	0.48	0.96	0.48	0.96	0	0	0
d_2	0	0.48	0	0	0	0.76	0.48	0
d_3	0	0	0	0.48	0	0	0	0.48
d_4	0	0	0	0	0	0.76	0.48	0.48

	one	was	a	race	horse	two	too	won
q	0	0	0	0	0	0	0	0.48

To find which document best matches the query, we compute the similarity of each document to the query. The document with the highest score is judged the most similar.

For the Vector Space Model, we use Cosine similarity:

$$\text{sim}(d, q) = \frac{d \cdot q}{\|d\|_2 \|q\|_2}$$

where the inner product between the document and query is:

$$d \cdot q = \sum_t w_{d,t} w_{q,t}$$

and the norm of each document and query is:

$$\|d\|_2 = \sqrt{\sum_t w_{d,t}^2} \qquad \|q\|_2 = \sqrt{\sum_t w_{q,t}^2}$$

For example, $d_1 \cdot q = 0\times0+0.48\times0+0.96\times0+0.48\times0+0.96\times0+0\times0+0\times0+0\times0+0\times0.48 = 0$. Also, $\|d_1\|_2 = \sqrt{0^2+0.48^2+0.96^2+0.48^2+0.96^2+0^2+0^2+0^2} = 1.52$

So we obtain:

	Norm	$d \cdot q$	Cosine
d_1	1.52	0	0
d_2	1.02	0	0
d_3	0.67	0.23	0.72
d_4	1.02	0.23	0.47
q	0.48		

Showing that d_3 is the best match for the query "one won" and d_4 is the second best match.

Language Model

Building the document models

To construct the Dirichlet smoothed language model, we must first compute the probability of each term in each document and the probability of each term in the collection.

We start with the set of term frequencies.

	one	was	a	race	horse	two	too	won
d_1	2	1	1	1	1	0	0	0
d_2	1	1	0	0	0	2	1	0
d_3	3	0	0	1	0	0	0	1
d_4	1	0	0	0	0	2	1	1

The probability of each word in a document $P(t|d)$ is the probability of choosing the word from all words in the document. It is simply the term frequency divided by the sum of term frequencies for the document (number of words in the document).

The set of $P(t|d)$ values are:

	one	was	a	race	horse	two	too	won
$P(t\|d_1)$	0.33	0.17	0.17	0.17	0.17	0	0	0
$P(t\|d_2)$	0.2	0.2	0	0	0	0.4	0.2	0
$P(t\|d_3)$	0.6	0	0	0.2	0	0	0	0.2
$P(t\|d_4)$	0.2	0	0	0	0	0.4	0.2	0.2

Note that each row sums to 1 (but the first row does not due to rounding).

The probability of a term given the collection is the same as the probability of a term given the document, except that we put all of the words from all documents into one document. If we combine all documents we get:

	one	was	a	race	horse	two	too	won
C	7	2	1	2	1	4	2	2

So the probability of each term, given the collection $P(t|C)$ is:

	one	was	a	race	horse	two	too	won
$P(t\|C)$	0.33	0.10	0.05	0.10	0.05	0.19	0.10	0.10

We combine $P(t|d)$ and $P(t|C)$ to obtain the term weights.

The mixing factor is $\lambda = n_d/(n_d + \alpha)$, where n_d is the number of words in document d and α is a constant. Let's set $\alpha = 1$.

	n_d	λ	$1 - \lambda$
d_1	6	0.86	0.14
d_2	5	0.83	0.17
d_3	5	0.83	0.17
d_4	5	0.83	0.17

The document model weights are $\lambda P(t|d) + (1 - \lambda)P(t|C)$.

	one	was	a	race	horse	two	too	won
d_1	0.33	0.16	0.15	0.16	0.15	0.02	0.01	0.01
d_2	0.22	0.18	0.01	0.02	0.01	0.37	0.18	0.02
d_3	0.56	0.02	0.01	0.18	0.01	0.03	0.02	0.18
d_4	0.22	0.02	0.01	0.02	0.01	0.37	0.18	0.18

Notice that these weights are the $P(t|d)$ probabilities, but they are pushed (smoothed) slightly towards the $P(t|C)$ probabilities.

Computing the similarity of each document to the query

Each document is now represented as a probability distribution over all terms.

For the query "one won", we ask "what is the probability of sampling the terms"one won" from the given document model. For d_1, the probability of sampling "one" is 0.33, the probability of sampling "won" is 0.01, therefore the probability of sampling both terms is the product (since we want "one" AND "won"), giving 0.0033. For this example, we have only a 8 words, and two query terms. For larger data sets, we would have thousands of words, and more query terms, so the probabilities would be very small. Rather than writing very small numbers, we take the log of the answer to make it more readable. So the score for d_1 is $\log(0.0033) = -5.71$.

The score for each document is:

	one	won	one × won	log(one × won)
d_1	0.33	0.01	0.0033	-5.71
d_2	0.22	0.02	0.0044	-5.43
d_3	0.56	0.18	0.1008	-2.29
d_4	0.22	0.18	0.0396	-3.23

So we find that d_3 is the best match for query "one won", with a score of -2.29, with d_4 being second best.