Adhoc retrieval from Legal Document(Consumer Law and Hindu Marriage & Diverse Law)

Dataset Pre-processing

Vocabulary is taken from here : http://www-01.sil.org/linguistics/wordlists/english/wordlist/wordsEn.txt.
Remove stop-words from vocab.
For each file :

1. Remove non-alphabet characters.
2. Tokenize and lemmatize.
3. For tokens not in vocab check for words stuck together(specific issue with the dataset) and resolve it using following heuristic: find largest prefix that is in vocab and then ,if still not found, try for suffix
4. Again remove tokens with len=2 and stop-words.

Dictionary creation

A seperate dictionary is created for every individual court.
Filter each of these dictionary separately by discarding words occurring in less than 5 documents and in more than 50%.
Merge all the dictionaries into one.

Models

TFIDF

Formula for unnormalized weight of term i in document j in a corpus of D documents:
weight_{i,j} = frequency_{i,j} * log_2(D / document_freq_{i})
We then normalize the whole vector to unit length.

BM25

The ranking function [edit]

BM25 is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of the inter-relationship between the query terms within a document (e.g., their relative proximity). It is not a single function, but actually a whole family of scoring functions, with slightly different components and parameters. One of the most prominent instantiations of the function is as follows.

Given a query Q, containing keywords $q_1, \ldots, q_n$, the BM25 score of a document D is:

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)},$$

where $f(q_i,D)$ is q_i's term frequency in the document D, $|D|$ is the length of the document D in words, and avgdl is the average document length in the text collection from which documents are drawn. k_1 and b are free parameters, usually chosen, in absence of an advanced optimization, as $k_1 \in [1.2, 2.0]$ and $b = 0.75$.[1] $\text{IDF}(q_i)$ is the IDF (inverse document frequency) weight of the query term q_i. It is usually computed as:

$$\text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5},$$

where N is the total number of documents in the collection, and $n(q_i)$ is the number of documents containing q_i.

Parameter chosen : K1 = 1.2, B = 0.75

Language Models

[A Study of Smoothing Methods for LanguageModels Applied to Information Retrieval ,CHENGXIANG ZHAI and JOHN LAFFERTY, Carnegie Mellon University]

One considers the probability of a query as being “generated” by a probabilistic model based on a document.

p(d|q) ∝ p(q|d)p(d)

p(d) is assumed to be uniform, and so does not affect document ranking.

The retrieval model reduces to the calculation of p(q|d), which is where language modeling comes in.

We use unigram model :

$$p(q \mid d) = \prod_{i=1}^{n} p(q_i \mid d)$$

Most smoothing methods make use of two distributions, a model ps(w|d) used for “seen” words that occur in the document, and a model pu(w|d) for “unseen” words that do not. The probability of a query q can be written in terms of these models as follows, where c(w; d) denotes the count of word w in d and n is the length of the query.

$$\begin{aligned}\log p(q \mid d) &= \sum_{i=1}^{n} \log p(q_i \mid d) \\ &= \sum_{i:c(q_i;d)>0} \log p_s(q_i \mid d) + \sum_{i:c(q_i;d)=0} \log p_u(q_i \mid d) \\ &= \sum_{i:c(q_i;d)>0} \log \frac{p_s(q_i \mid d)}{p_u(q_i \mid d)} + \sum_{i=1}^{n} \log p_u(q_i \mid d)\end{aligned}$$

The probability of an unseen word is typically taken as being proportional to the general frequency of the word, e.g., as computed using the document collection. So, let us assume that $p_u(q_i \mid d) = \alpha_d \, p(q_i \mid \mathcal{C})$, where α_d is a document-dependent constant and $p(q_i \mid \mathcal{C})$ is the collection language model. Now we have

$$\log p(q \mid d) = \sum_{i:c(q_i;d)>0} \log \frac{p_s(q_i \mid d)}{\alpha_d \, p(q_i \mid \mathcal{C})} + n \log \alpha_d + \sum_{i=1}^{n} \log p(q_i \mid \mathcal{C}) \qquad (6)$$

Note that the last term on the righthand side is independent of the document d, and thus can be ignored in ranking.

αd is a document-dependent constant that is related to how much probability mass will be allocated to unseen words according to the particular smoothing method used.

p(qi|C) plays a role very similar to the well-known IDF.

nlogαd can be thought of playing the role of document length normalization.

$$p_{ml}(w \mid d) = \frac{c(w;d)}{\sum_{w' \in V} c(w';d)}$$

Method	$p_s(w \mid d)$	α_d	Parameter
Jelinek-Mercer	$(1-\lambda)\, p_{ml}(w \mid d) + \lambda p(w \mid \mathcal{C})$	λ	λ
Dirichlet	$\frac{c(w;d) + \mu\, p(w \mid \mathcal{C})}{\lvert d\rvert + \mu}$	$\frac{\mu}{\lvert d\rvert + \mu}$	μ

Language Model - Jelinek-Mercer Smoothing

5.1 Jelinek-Mercer smoothing

When using the Jelinek-Mercer smoothing method with a fixed λ, we see that the parameter α_d in our ranking function (see Section 2) is the same for all documents, so the length normalization term is a constant. This means that the score can be interpreted as a sum of weights over each matched term. The term weight is $\log(1+(1-\lambda)p_{ml}(q_i|d)/(\lambda p(q_i \mid \mathcal{C})))$.

Parameter chosen : Lambda = 0.7

Language Model - Dirichlet Prior Smoothing

The term wieght is:

$$\log(1 + c(q_i; d)/(\mu p(q_i \mid \mathcal{C})))$$

and an additional “|q| * log(MU/(|d|+ MU)” term added to the total score. |q| is the query length and |d| is the document length.

Parameter chosen : Mu = 2000

Models summary

1. TFIDF
2. BM25 : K1 = 1.2, B = 0.75
3. Language Model - Jelinek-Mercer Smoothing : Lambda = 0.7
4. Language Model - Dirichlet Prior Smoothing : Mu = 2000

Results

Fscore

Harmonic mean of precision and recall.

Average precision

$$\text{AveP} = \frac{\sum_{k=1}^{n}(P(k) \times \text{rel}(k))}{\text{number of relevant documents}}$$

Where k is the rank in the sequence of retrieved documents, n is the number of retrieved documents, P(k) is the precision at cut-off k in the list and rel(k) is an indicator function equaling 1 if the item at rank k is a relevant document, zero otherwise.

Mean Average Precision

Mean average precision for a set of queries is the mean of the average precision scores for each query.

$$\text{MAP} = \frac{\sum_{q=1}^{Q} \text{AveP(q)}}{Q}$$

where *Q* is the number of queries.

Evaluations performed

k = 1, 5 ,10, 25, 50, 100, 500, 1000.

For each query

1. Precision, Recall and FScore@k for Relevance = 1, 0 and 1 & 0 both.
2. Average Precision@k for Relevance = 1, 0 and 1 & 0 both.

Mean Average precison@k for Relevance = 1, 0 and 1 & 0 both.