Text Retrieval:
Probabilistic Relevance and Latent Semantic Indexing

RAHUL SIMHA
Department of Computer Science
The George Washington University

The goal of these notes is to describe the key ideas in two approaches to text retrival:

- *Probabilistic relevance* – the idea of identifying relevant documents by the "probability that a document is relevant to a query".
- *Latent Semantic Indexing* – the idea of treating documents as vectors and "finding the closest vector" to a query vector.

For a quick and dirty overview of basic linear algebra see [5] – but you'd be better off following that up with Strang's outstanding linear algebra textbook [6]. The presentation of these ideas follows that of the Information Retrieval textbook of Manning et al [3].

1 Probabilistic Relevance

- Define the following:
  - There are m terms in the universe.
  - d = a document
  - Represent d by vector $\mathbf{d} = (d_1, \ldots, d_m)$ where $d_i = 1$ if term i is in the document.
  - q = a query
  - Represent the query as a vector $\mathbf{q} = (q_1, \ldots, q_m)$ where $q_i = 1$ if term i is in the query.
  - $R = R(d, q)$ = relevance of a document d to query q.
  - In the binary relevance model: $R = 1$ or $R = 0$.
- We will be interested in the conditional probability

  $$P\left[R = 1|\mathbf{d}, \mathbf{q}\right]$$

  which asks "given a document $\mathbf{d}$ and query $\mathbf{q}$, what is the probability that the relevance is 1?"

- From Bayes' rule:

$$P[R=1|\mathbf{d},\mathbf{q}] = \frac{P[\mathbf{d}|R=1,\mathbf{q}]\,P[R=1|\mathbf{q}]}{P[\mathbf{d}|\mathbf{q}]}$$

- At first, these terms on the right look strange and hard to evaluate, but we'll make some simplifying assumptions.

- The first one is: we'll assume $P[R=1|\mathbf{q}]$ is independent of documents.

  ⇒this is merely some guess about how often the system returns relevant queries

- To help in canceling out terms, we'll also compute

$$P[R=0|\mathbf{d},\mathbf{q}] = \frac{P[\mathbf{d}|R=0,\mathbf{q}]\,P[R=0|\mathbf{q}]}{P[\mathbf{d}|\mathbf{q}]}$$

- Define the relevance ratio

$$\rho = \frac{P[R=1|\mathbf{d},\mathbf{q}]}{P[R=0|\mathbf{d},\mathbf{q}]}$$

  Thus, the higher ρ is, the "better" the match between the document and the query.

  What's nice about this definition is that it'll let us get rid of the common denominator $P[\mathbf{d}|\mathbf{q}]$.

- Substituting from the Bayes' rule expansion,

$$\rho = \frac{P[\mathbf{d}|R=1,\mathbf{q}]\,P[R=1|\mathbf{q}]}{P[\mathbf{d}|R=0,\mathbf{q}]\,P[R=0|\mathbf{q}]}$$

- Now for the next simplifying assumption: the ratio

$$\frac{P[R=1|\mathbf{q}]}{P[R=0|\mathbf{q}]}$$

  does not involve the document and can be assumed to be some fixed "system" constant.

  ⇒We can estimate this offline and use the estimate .

  Accordingly, let's call this constant α and write

$$\rho = \alpha \frac{P[\mathbf{d}|R=1,\mathbf{q}]}{P[\mathbf{d}|R=0,\mathbf{q}]}$$

- The next assumption will involve independence: we will write

$$P[\mathbf{d}|R=1,\mathbf{q}] = \prod_{i=1}^{m} P[d_i|R=1,\mathbf{q}]$$

which assumes that the terms occur independently in documents.

This is patently not true, but we'll hope that the resulting search is nonetheless effective.

- Thus, after these assumptions

$$\rho = \alpha \prod_{i=1}^{m} \frac{P[d_i|R=1,\mathbf{q}]}{P[d_i|R=0,\mathbf{q}]}$$

- The next trick is going to separate the product terms by $d_i = 1$ and $d_i = 0$ so that

$$\rho = \alpha \prod_{i:x_i=1}^{m} \frac{P[d_i=1|R=1,\mathbf{q}]}{P[d_i=1|R=0,\mathbf{q}]} \prod_{i:x_i=0}^{m} \frac{P[d_i=0|R=1,\mathbf{q}]}{P[d_i=0|R=0,\mathbf{q}]}$$

- Define, for shorthand
  - $p_i = P[d_i=1|R=1,\mathbf{q}]$.
  - $u_i = P[d_i=1|R=0,\mathbf{q}]$.

  Then,
  - $P[d_i=0|R=1,\mathbf{q}] = 1 - p_i$.
  - $P[d_i=0|R=0,\mathbf{q}] = 1 - u_i$.

- The next simplifying assumption: for terms that do not occur in the query, assume that they are equally likely to occur in relevant vs. non-relevant documents.
  $\Rightarrow$if $q_i = 0$ then $p_i = u_i$.

- Accordingly, we'll separate out the terms where $q_i = 1$ vs. $q_i = 0$:

$$\begin{aligned}
\rho &= \alpha \prod_{i:x_i=1,q_i=1} \frac{p_i}{u_i} \prod_{i:x_i=0,q_i=1} \frac{1-p_i}{1-u_i} \prod_{i:x_i=1,q_i=0} \frac{p_i}{u_i} \prod_{i:x_i=0,q_i=0} \frac{1-p_i}{1-u_i} \\
&= \alpha \prod_{i:x_i=1,q_i=1} \frac{p_i}{u_i} \prod_{i:x_i=0,q_i=1} \frac{1-p_i}{1-u_i}
\end{aligned}$$

- Consider for a moment the product

$$\prod_{i:q_i=1} \frac{1-p_i}{1-u_i}$$

We'll expand this as

$$\prod_{i:x_i=1,q_i=1} \frac{1-p_i}{1-u_i} \prod_{i:x_i=0,q_i=1} \frac{1-p_i}{1-u_i}$$

Thus, isolating the second term on one side:

$$\prod_{i:x_i=0,q_i=1} \frac{1-p_i}{1-u_i} = \prod_{i:x_i=1,q_i=1} \frac{1-u_i}{1-p_i} \prod_{i:q_i=1} \frac{1-p_i}{1-u_i}$$

The right-hand-side is in the earlier expression for ρ. We'll substitute the left-hand-side into that expression:

$$\begin{aligned} \rho &= \alpha \prod_{i:x_i=1,q_i=1} \frac{p_i}{u_i} \prod_{i:x_i=0,q_i=1} \frac{1-p_i}{1-u_i} \\ &= \alpha \prod_{i:x_i=1,q_i=1} \frac{p_i}{u_i} \prod_{i:x_i=1,q_i=1} \frac{1-u_i}{1-p_i} \prod_{i:q_i=1} \frac{1-p_i}{1-u_i} \\ &= \alpha \prod_{i:x_i=1,q_i=1} \frac{p_i(1-u_i)}{u_i(1-p_i)} \prod_{i:q_i=1} \frac{1-p_i}{1-u_i} \end{aligned}$$

- Examine the last term. This is again independent of document and is a "system constant" that we'll roll into α.

  Thus,

$$\rho = \alpha \prod_{i:x_i=1,q_i=1} \frac{p_i(1-u_i)}{u_i(1-p_i)}$$

- The only remaining thing is to estimate p_i and u_i.
- Recall that $u_i = P\left[d_i = 1 | R = 0, \mathbf{q}\right]$.
  - Given irrelevance ($R = 0$), what are the chances that a particular term appears in the document ($d_i = 1$)?
  - If we have a large document collection, there is no apriori reason to believe that terms prefer certain documents relevant to the query.

  - Accordingly, a reasonable assumption is

$$u_i \approx \frac{\#\text{ doc's in which term } i \text{ occurs}}{\#\text{ doc's}}$$

  - This can be estimated offline for every term i.

- This leaves $p_i = P\left[d_i = 1 | R = 1, \mathbf{q}\right]$.
  - This asks: if you know a document is relevant to a query, what are the chances that term i played a role in the relevance?
  - This boils down to identifying the key terms in a document.

- There are many heuristics for estimating p_i:
  - Use frequency of occurence of term i.
  - Use some offline queries or query history to estimate p_i.
  - Build a model based on small samples.

- Once we are able to compute ρ for any document, we simply return a ranked list of documents, sorted in decreasing order of ρ.

- History:
  - The probabilistic-relevance model was first proposed by Maron and Kuhn in 1960 [4].
  - Since then it has been refined by several people.
  - One of the most popular refinements has been the Okapi-BM25 algorithm developed by Robertson and colleagues [2].

2 Latent Semantic Indexing

- Let's start by reviewing the vector space model:
  - Suppose there are m terms.
  - Each document $\mathbf{d}_k$ can be written as a vector $\mathbf{d}_k = (d_{k,1}, \ldots, d_{k,m})$.
  - In the binary model $d_{k,i} = 1$ or 0 depending on whether the i-th term is in the document or not.
  - One can use real numbers that reflect "importance" of the i-th term to the document.

- For example, it's common to combine the term's occurence-count in the document vs. the term's occurence in the whole document set, e.g.,

$$d_{k,i} = \frac{\#\text{ occurences of term } i \text{ in } \mathbf{d}_k}{\beta\#\text{ occurences of term } i \text{ in all documents}}$$

  where β is needed because the other denominator term can be quite large.
- Here, the idea is, if a term occurs in too many documents, it's not going to be useful in discriminating between them.
- Similarly, the query is also a vector of this kind: $\mathbf{q} = (q_1, \ldots, q_m)$.

• The cosine distance between a query and the k-th document is:

$$R(\mathbf{q}, \mathbf{d}_k) = \frac{\mathbf{q}\mathbf{d}_k}{|\mathbf{q}||\mathbf{d}_k|}$$

- Notice that R is always between 0 and 1 (assuming positive vector components).
- The closer to 1, the smaller the angle
  $\Rightarrow$the more relevant the document
- Thus, one can easily rank documents by their R value.

• Two problems with cosine products:

- For a large number of terms (e.g., 100,000 words), a single dot-product can take time.
- Most of the dot-product computations are zero.
- The vector-space approach does not take synonyms into account.

• The Latent Semantic Approach [1] is an attempt to solve both problems with one technique.

• To start with, let's define the $m \times n$ term-document matrix $\mathbf{A}$ as a matrix with the documents as columns:

- m rows, one per term.
- n columns, one per document.

- The k-th column is the document vector $\mathbf{d}_k$.
- Thus, $\mathbf{A}_{ij}$ = the "importance" of term i in document j.

- Note:
  - The columns are very long ($m = 100,000$ or more).
  - The matrix is quite sparse.

- Suppose there was a way to reduce the length of each document vector:
  - Suppose we could reduce $\mathbf{d}_k$'s length to 100, and that this was all non-zero (i.e., useful content).
  - Then, dot-products would take less time and be useful.
  - There are many techniques for dimension-reduction.
  - For example: use the 100 most important rows of A.
    $\Rightarrow$But this means selecting 100 terms only .
  - Ideally, we'd like to combine terms in some way to "squish" the matrix down into a smaller space.
  - This raises the problem of also "squishing" the query.

  The Singular Value Decomposition (SVD) offers one solution to this dimension reduction.

- The SVD takes an $m \times n$ matrix $\mathbf{A}$ and a number k and creates three matrices:
  - An $m \times k$ matrix $\mathbf{U_k}$.
  - A $k \times k$ matrix $\mathbf{\Sigma_k}$.
  - An $n \times k$ matrix $\mathbf{V_k}$

  These matrices can be multiplied as follows to give a matrix A_k:

$$\mathbf{A_k} = \mathbf{U_k}\mathbf{\Sigma_k}\mathbf{V_k}^T$$

- The idea is to approximate $\mathbf{A}$ using $\mathbf{A_k}$ with small k, e.g., $k = 100$.
  - Notice, however that $\mathbf{A_k}$ is $m \times n$, the same size as $\mathbf{A}$.
  - The savings will come when we handle queries.

- What is known about this approximation and whether it's possible?
  - It is known that $\mathbf{A_k}$ has rank k.
    $\Rightarrow k$ LI columns (or rows) .
  - It is known that $\mathbf{A_k}$ is the "best" rank-k approximation to $\mathbf{A}$.
    $\Rightarrow$This is with the so-called Frobenius norm .
- How does one get this matrix $\mathbf{A_k}$?
  - The SVD theorem states that any matrix $\mathbf{A}$ can be decomposed as
    $$\mathbf{A} = \mathbf{U\Sigma V}^T$$
    where the columns of $\mathbf{U}$ are the eigenvectors of $\mathbf{AA^T}$, the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{A^TA}$, and Σ is a diagonal matrix.
  - $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r, \sigma_{r+1}, \ldots, \sigma_n)$ are the so-called *singular values* or "importance weights".
    $\Rightarrow$Some of these will be small or zero, and can be ignored .
  - If $\sigma_{r+1} = \sigma_{r+2} = \ldots = \sigma_n = 0$ then
    $$\mathbf{A} = \mathbf{U_r \Sigma_r V_r}^T$$
    That is, those parts of the matrices can be thrown away.
  - Now, one can pick $k < r$ so that we get an even smaller decomposition.
- Of the resulting compressed version, the columns of $\mathbf{V_k}$ are the "compressed" documents.
  $\Rightarrow$These are size-k vectors in a different space of "coalesced documents"
- A query $\mathbf{q}$ needs to be reduced in dimension (to size k) as well.
  - Define
    $$\mathbf{q}' = \mathbf{qU_k\Sigma_k}^{-}1$$
  - This produces a "reduced" document that can be compared to the reduced columns of V_k.
  - The documents are then returned in order of cosine-closeness.

References

[1] S.Deerwester, S.T.Dumais, G.W.Furnas, T.K.Landauer and R.Harshman. Indexing by latent semantic analysis. *J.Am.Soc.Info.Sci.*, vol.41, no. 6, pp.391-407, 1990.

[2] K.S.Jones, S.Walker, and S.E.Robertson. A Probabilistic Model of Information Retrieval: Development and Comparative Experiments. *Information Processing and Management*, 36(6):779-840. 2000.

[3] C.D.Manning, P.Raghavan and H.Schutze. *Introduction to Information Retrieval*, Cambridge University Press. 2008. http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

[4] M.E.Maron and J.Kuhns. On relevance, probabilistic indexing and information retrieval. *J. ACM*, 7:3, pp.216-244, 1960.

[5] R.Simha. Elementary linear algebra – a quick overview. http://www.seas.gwu.edu/˜simhaweb/misc/linearalgebra

[6] G.Strang. *Linear Algebra and Its Applications*, Brooks-Cole, 1988.