CT4100

Information Retrieval

Name: Andrew Hayes
E-mail: a.hayes18@universityofgalway.ie
Student ID: 21321503

2024–10–04

Contents

1 Introduction

1.1 Lecturer Contact Details

- Colm O'Riordan.
- `colm.oriordan@universityofgalway.ie`.

1.2 Motivations

- To study/analyse techniques to deal suitably with the large amounts (& types) of information.
- Emphasis on research & practice in Information Retrieval.

1.3 Related Fields

- Artificial Intelligence.
- Database & Information Systems.
- Algorithms.
- Human-Computer Interaction.

1.4 Recommended Texts

- *Modern Information Retrieval* – Riberio-Neto & Baeza-Yates (several copies in library).
- *Information Retrieval* – Grossman.
- *Introduction to Information Retrieval* – Christopher Manning.
- Extra resources such as research papers will be recommended as extra reading.

1.5 Grading

- Exam: 70%.
- Assignment 1: 30%.
- Assignment 2: 30%.

There will be exercise sheets posted for most lecturers; these are not mandatory and are intended as a study aid.

1.6 Introduction to Information Retrieval

Information Retrieval (IR) deals with identifying relevant information based on users' information needs, e.g. web search engines, digital libraries, & recommender systems. It is finding material (usually documents) of an unstructured nature that satisfies an information need within large collections (usually stored on computers).

2 Information Retrieval Models

2.1 Introduction to Information Retrieval Models

Data collections are well-structured collections of related items; items are usually atomic with a well-defined interpretation. Data retrieval involves the selection of a fixed set of data based on a well-defined query (e.g., SQL, OQL).

Information collections are usually semi-structured or unstructured. Information Retrieval (IR) involves the retrieval of documents of natural language which is typically not structured and may be semantically ambiguous.

2.1.1 Information Retrieval vs Information Filtering

The main differences between information retrieval & information filtering are:

- The nature of the information need.
- The nature of the document set.

Other than these two differences, the same models are used. Documents & queries are represented using the same set of techniques and similar comparison algorithms are also used.

2.1.2 User Role

In traditional IR, the user role was reasonably well-defined in that a user:

- Formulated a query.
- Viewed the results.
- Potentially offered feedback.
- Potentially reformulated their query and repeated steps.

In more recent systems, with the increasing popularity of the hypertext paradigm, users usually intersperse browsing with the traditional querying. This raises many new difficulties & challenges.

2.2 Pre-Processing

Document pre-processing is the application of a set of well-known techniques to the documents & queries prior to any comparison. This includes, among others:

- **Stemming:** the reduction of words to a potentially common root. The most common stemming algorithms are Lovin's & Porter's algorithms. E.g. *computerisation*, *computing*, *computers* could all be stemmed to the common form *comput*.
- **Stop-word removal:** the removal of very frequent terms from documents, which add little to the semantics of meaning of the document.
- **Thesaurus construction:** the manual or automatic creation of thesauri used to try to identify synonyms within the documents.

Representation & comparison technique depends on the information retrieval model chosen. The choice of feedback techniques is also dependent on the model chosen.

2.3 Models

Retrieval models can be broadly categorised as:

- Boolean:
  - Classical Boolean.
  - Fuzzy Set approach.
  - Extended Boolean.
- Vector:
  - Vector Space approach.
  - Latent Semantic indexing.
  - Neural Networks.

- Probabilistic:
  - Inference Network.
  - Belief Network.

We can view any IR model as being comprised of:

- D is the set of logical representations within the documents.
- Q is the set of logical representations of the user information needs (queries).
- F is a framework for modelling representations (D & Q) and the relationship between D & Q.
- R is a ranking function which defines an ordering among the documents with regard to any query q.

We have a set of index terms:

$$t_1, \ldots, t_n$$

A **weight** $w_{i,j}$ is assigned to each term t_i occurring in the d_j. We can view a document or query as a vector of weights:

$$\vec{d_j} = (w_1, w_2, w_3, \ldots)$$

2.4 Boolean Model

The **Boolean model** of information retrieval is based on set theory & Boolean algebra. A query is viewed as a Boolean expression. The model also assumes terms are present or absent, hence term weights $w_{i,j}$ are binary & discrete, i.e., $w_{i,j}$ is an element of $\{0, 1\}$.

Advantages of the Boolean model include:

- Clean formalism.
- Widespread & popular.
- Relatively simple

Disadvantages of the Boolean model include:

- People often have difficulty formulating expressions, harbours some difficulty in use.
- Documents are considered either relevant or irrelevant; no partial matching allowed.
- Poor performance.
- Suffers badly from natural language effects of synonymy etc.
- No ranking of results.
- Terms in a document are considered independent of each other.

2.4.1 Example

$$q = t_1 \wedge (t_2 \vee (\neg t_3))$$

```
1 q = t1 AND (t2 OR (NOT t3))
```

This can be mapped to what is termed **disjunctive normal form**, where we have a series of disjunctions (or logical ORs) of conjunctions.

$$q = 100 \vee 110 \vee 111$$

If a document satisfies any of the components, the document is deemed relevant and returned.

2.5 Vector Space Model

The **vector space model** attempts to improve upon the Boolean model by removing the limitation of binary weights for index terms. Terms can have non-binary weights in both queries & documents. Hence, we can represent the documents & the query as n-dimensional vectors.

$$\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{n,j})$$
$$\vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{n,q})$$

We can calculate the similarity between a document & a query by calculating the similarity between the vector representations of the document & query by measuring the cosine of the angle between the two vectors.

$$\vec{a} \cdot \vec{b} = \mid \vec{a} \mid\mid \vec{b} \mid \cos(\vec{a}, \vec{b})$$

$$\Rightarrow \cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\mid \vec{a} \mid\mid \vec{b} \mid}$$

We can therefore calculate the similarity between a document and a query as:

$$\text{sim}(q, d) = \cos(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{\mid \vec{q} \mid\mid \vec{d} \mid}$$

Considering term weights on the query and documents, we can calculate similarity between the document & query as:

$$\text{sim}(q, d) = \frac{\sum_{i=1}^{N}(w_{i,q} \times w_{i,d})}{\sqrt{\sum_{i=1}^{N}(w_{i,q})^2} \times \sqrt{\sum_{i=1}^{N}(w_{i,d})^2}}$$

Advantages of the vector space model over the Boolean model include:

- Improved performance due to weighting schemes.
- Partial matching is allowed which gives a natural ranking.

The primary disadvantage of the vector space model is that terms are considered to be mutually independent.

2.5.1 Weighting Schemes

We need a means to calculate the term weights in the document and query vector representations. A term's frequency within a document quantifies how well a term describes a document; the more frequently a term occurs in a document, the better it is at describing that document and vice-versa. This frequency is known as the **term frequency** or **tf factor**.

If a term occurs frequently across all the documents, that term does little to distinguish one document from another. This factor is known as the **inverse document frequency** or **idf-frequency**. Traditionally, the most commonly-used weighting schemes are know as **tf-idf** weighting schemes.

For all terms in a document, the weight assigned can be calculated as:

$$w_{i,j} = f_{i,j} \times \log\left(\frac{N}{N_i}\right)$$

where

- $f_{i,j}$ is the (possibly normalised) frequency of term t_i in document d_j.
- N is the number of documents in the collection.
- N_i is the number of documents that contain term t_i.

3 Evaluation of IR Systems

When evaluating an IR system, we need to consider:

- The **functional requirements**: whether or not the system works as intended. This is done with standard testing techniques.
- The **performance:**
  - Response time.
  - Space requirements.
  - Measure by empirical analysis, efficiency of algorithms & data structures for compression, indexing, etc.
- The **retrieval performance:** how useful is the system? IR is a highly empirical discipline and there is a long history of the evaluation of retrieval performance. This is less of an issue in data retrieval systems wherein perfect matching is possible as there exists a correct answer.

3.1 Test Collections

Evaluation of IR systems is usually based on a reference **test collection** involving human evaluations. The test collection usually comprises:

- A collection of documents D.
- A set of information needs that can be represented as queries.
- A list of relevance judgements for each query-document pair.

Issues with using test collections include:

- It can be very costly to obtain relevance judgements.
- Crowd sourcing.
- Pooling approaches.
- Relevance judgements don't have to be binary.
- Agreement among judges.

TREC (Text REtrieval Conference) provides a means to empirically test the performance of systems in different domains by providing *tracks* consisting of a data set & test problems. These tracks include:

- **Ad-hoc retrieval:** different tracks have been proposed to test ad-hoc retrieval including the Web track (retrieval on web corpora) and the Million Query track (large number of queries).
- **Interactive Track**: users interact with the system for relevance feedback.
- **Contextual Search:** multiple queries over time.
- **Entity Retrieval:** the task is to retrieve entities (people, places, organisations).
- **Spam Filtering:** identifying & filtering out non-relevant or harmful content such as email spam.
- **Question Answering (QA):** the goal is to retrieve precise answers to user questions rather than returning entire documents.
- **Cross-Language Retrieval:** the goal is to retrieve relevant documents in a different language from the query. Requires machine translation.
- **Conversational IR:** retrieving information in conversational IR systems.

- **Sentiment Retrieval:** emphasis on identifying opinions & sentiments.
- **Fact Checking:** misinformation track.
- **Domain-Specific Retrieval:** e.g., genomic data.
- Summarisation Tasks.

Relevance is assessed for the information need and not the query. Because tuning & optimisation can occur for many IR systems, it is considered good practice to tune on one collection and then test on another.

Interaction with an IR system may be a one-off query or an interactive session. For the former, *quality* of the returned set is the important metric, while for interactive systems other issues have to be considered: duration of the session, user effort required, etc. These issues make evaluation of interactive sessions more difficult.

3.2 Precision & Recall

The most commonly used metrics are **precision** & **recall**.

3.2.1 Unranked Sets

Given a set D and a query Q, let R be the set of documents relevant to Q. Let A be the set actually returned by the system.

- **Precision** is defined as $\frac{|R \cap A|}{|A|} = \frac{\text{relevant retrieved documents}}{\text{all retrieved documents}}$, i.e. what fraction of the retrieved documents are relevant.
- **Recall** is defined as $\frac{|R \cap A|}{|R|} = \frac{\text{relevant retrieved documents}}{\text{all relevant documents}}$, i.e. what fraction of the relevant documents were returned.

Having two separate measures is useful as different IR systems may have different user requirements. For example, in web search precision is of the greatest importance, but in the legal domain recall is of the greatest importance.

There is a trade-off between the two measures; for example, by returning every document in the set, recall is maximised (because all relevant documents will be returned) but precision will be poor (because many irrelevant documents will be returned). Recall is non-decreasing as the number of documents returned increases, while precision usually decreases as the number of documents returned increases.

	Relevant	**Non-Relevant**
Relevant	True Positive (TP)	False Negative (FN)
Non-Relevant	False Positive (FP)	True Negative (TN)

Table 1: Confusion Matrix of True/False Positives & Negatives

$$\text{Precision } P = \frac{tp}{tp + fp} = \frac{\text{true positives}}{\text{true positives + false positives}}$$

$$\text{Recall } R = \frac{tp}{tp + fn} = \frac{\text{true positives}}{\text{true positives + false negatives}}$$

The **accuracy** of a system is the fraction of these classifications that are correct:

$$\text{Accuracy} = \frac{tp + tn}{tp + fp + fn + tn}$$

Accuracy is a commonly used evaluation measure in machine learning classification work, but is not a very useful measure in IR; for example, when searching for relevant documents in a very large set, the number of irrelevant documents is usually much higher than the number of relevant documents, meaning that a high accuracy score is attainable by getting true negatives by discarding most documents, even if there aren't many true positives.

There are also many single-value measures that combine precision & recall into one value:

- F-measure.
- Balanced F-measure.

3.2.2 Evaluation of Ranked Results

In IR, returned documents are usually ranked. One way of evaluating ranked results is to use **Precision-Recall plots**, wherein precision is typically plotted against recall. In an ideal system, we would have a precision value of 1 for a recall value of 1, i.e., all relevant documents have been returned and no irrelevant documents have been returned.

Example

Given $|D| = 20$ & $|R| = 10$ and a ranked list of length 10, let the returned ranked list be:

$$\mathbf{d_1}, \mathbf{d_2}, d_3, \mathbf{d_4}, d_5, d_6, \mathbf{d_7}, d_8, d_9, d_{10}$$

where those in items in bold are those that are relevant.

- Considering the list as far as the first document: Precision = 1, Recall = 0.1.
- As far as the first two documents: Precision = 1, Recall = 0.2.
- As far as the first three documents: Precision = 0.67, Recall = 0.2.

We usually plot for recall values = 10% ... 90%.

We typically calculate precision for these recall values over a set of queries to get a truer measure of a system's performance:

$$P(r) = \frac{1}{N} \sum_{i=1}^{N} P_i(r)$$

Advantages of Precision-Recall include:

- Widespread use.
- It gives a definable measure.
- It summarises the behaviour of an IR system.

Disadvantages of Precision-Recall include:

- It's not always possible to calculate the recall measure effective of queries in batch mode.
- Precision & recall graphs can only be generated when we have ranking.
- They're not necessarily of interest to the user.

Single-value measures for evaluating ranked results include:

- Evaluating precision when every new document is retrieved and averaging precision values.
- Evaluating precision when the first relevant document is retrieved.
- R-precision: calculate precision when the final document has been retrieved.
- Precision at k (P@k).
- Mean Average Precision (MAP).

Precision histograms are used to compare two algorithms over a set of queries. We calculate the R-precision (or possibly another single summary statistic) of two systems over all queries. The difference between the two are plotted for each of the queries.

3.3 User-Oriented Measures

Let D be the document set, R be the set of relevant documents, A be the answer set returned to the users, and U be the set of relevant documents previously known to the user. Let AU be the set of returned documents previously known to the user.

$$\text{Coverage} = \frac{|AU|}{|U|}$$

Let *New* refer to the set of relevant documents returned to the user that were previously unknown to the user. We can define **novelty** as:

$$\text{Novelty} = \frac{|\text{New}|}{|\text{New}| + |AU|}$$

The issues surrounding interactive sessions are much more difficult to assess. Much of the work in measuring user satisfaction comes from the field of HCI. The usability of these systems is usually measured by monitoring user behaviour or via surveys of the user's experience. Another closely related area is that of information visualisation: ow best to represent the retrieved data for a user etc.

4 Weighting Schemes

4.1 Re-cap

The **vector space model** attempts to improve upon the Boolean model by removing the limitation of binary weights for index terms. Terms can have a non-binary value both in queries & documents. Hence, we can represent documents & queries as n-dimensional vectors:

$$\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{n,j})$$
$$\vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{n,q})$$

We can calculate the similarity between a document and a query by calculating the similarity between the vector representations. We can measure this similarity by measuring the cosine of the angle between the two vectors. We can derive a formula for this by starting with the formula for the inner product (dot product) of two vectors:

$$a \cdot b = |a||b| \cos(a, b) \tag{1}$$
$$\Rightarrow \cos(a, b) = \frac{a \cdot b}{|a||b|} \tag{2}$$

We can therefore calculate the similarity between a document and a query as:

$$\text{sim}(\vec{d_j}, \vec{q}) = \frac{d_j \cdot q}{|d_j||q|}$$
$$\Rightarrow \text{sim}(\vec{d_j}, \vec{q}) = \frac{\sum_{i=1}^{n} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{n} w_{i,q}^2}}$$

We need a means to calculate the term weights in the document & query vector representations. A term's frequency within a document quantifies how well a term describes a document. The more frequent a term occurs in a document, the better it is at describing that document and vice-versa. This frequency is known as the **term frequency** or **tf factor**.

However, if a term occurs frequently across all the documents, then that term does little to distinguish one document from another. This factor is known as the **inverse document frequency** or **idf-frequency**. The most commonly used weighting schemes are known as **tf-idf** weighting schemes For all terms in a document, the weight assigned can be calculated by:

$$w_{i,j} = f_{i,j} \times \log \frac{N}{n_i}$$

where $f_{i,j}$ is the normalised frequency of term t_i in document d_j, N is the number of documents in the collection, and n_i is the number of documents that contain the term t_i.

A similar weighting scheme can be used for queries. The main difference is that the tf & idf are given less credence, and all terms have an initial value of 0.5 which is increased or decreased according to the tf-idf across the document collection (Salton 1983).

4.2 Text Properties

When considering the properties of a text document, it is important to note that not all words are equally important for capturing the meaning of a document and that text documents are comprised of symbols from a finite alphabet.

Factors that affect the performance of information retrieval include:

- What is the distribution of the frequency of different words?
- How fast does vocabulary size grow with the size of a document collection?

These factors can be used to select appropriate term weights and other aspects of an IR system.

4.2.1 Word Frequencies

A few words are very common, e.g. the two most frequent words "the" & "of" can together account for about 10% of word occurrences. Most words are very rare: around half the words in a corpus appear only once, which is known as a "heavy tailed" or Zipfian distribution.

Zipf's law gives an approximate model for the distribution of different words in a document. It states that when a list of measured values is sorted in decreasing order, the value of the n^{th} entry is approximately inversely proportional to n. For a word with rank r (the numerical position of the word in a list sorted in by decreasing frequency) and frequency f, Zipf's law states that $f \times r$ will equal a constant. It represents a power law, i.e. a straight line on a log-log plot.

$$\text{word frequency} \propto \frac{1}{\text{word rank}}$$

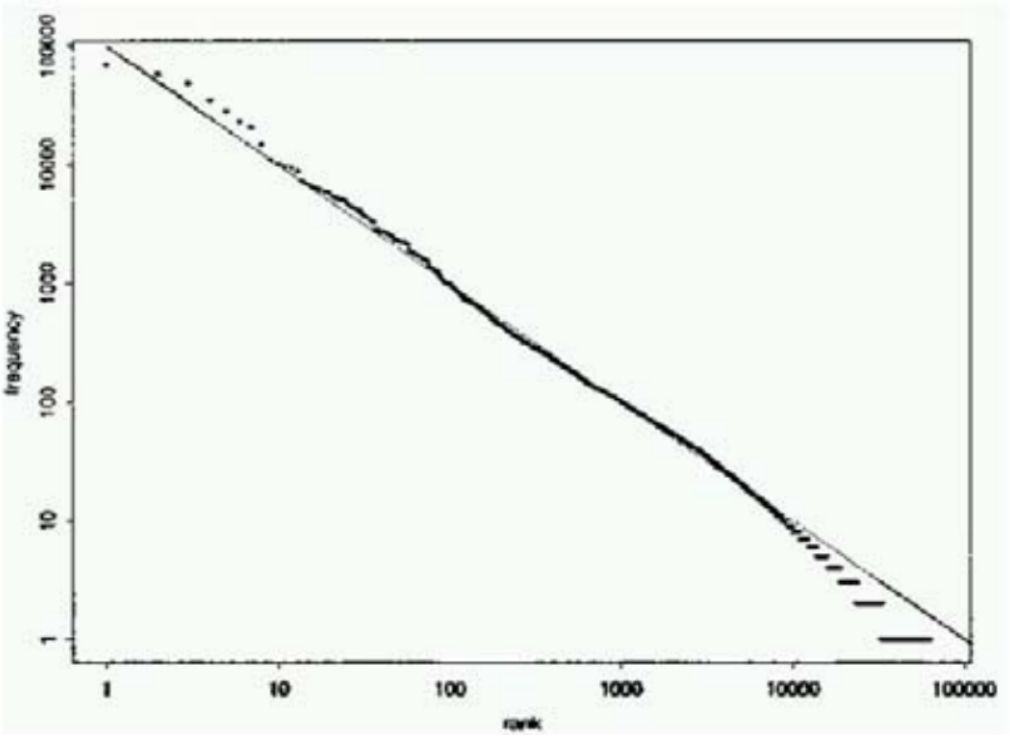

Figure 1: Zipf's Law Modelled on the Brown Corpus

As can be seen above, Zipf's law is an accurate model excepting the extremes.

4.3 Vocabulary Growth

The manner in which the size of the vocabulary increases with the size of the document collection has an impact on our choice of indexing strategy & algorithms. However, it is important to note that the size of a vocabulary is not really bounded in the real world due to the existence of mispellings, proper names etc., & document identifiers.

If V is the size of the vocabulary and n is the length of the document collection in word occurrences, then

$$V = K \cdot n^{\beta}, \quad 0 < \beta < 1$$

where K is a constant scaling factor that determines the initial vocabulary size of a small collection, usually in the range 10 to 100, and β is constant controlling the rate at which the vocabulary size increases usually in the range 0.4 to 0.6.

4.4 Weighting Schemes

The quality of performance of an IR system depends on the quality of the weighting scheme; we want to assign high weights to those terms with a high resolving power. tf-idf is one such approach wherein weight is increased for frequently occurring terms but decreased again for those that are frequent across the collection. The "bag of words" model is usually adopted, i.e., that a document can be treated as an unordered collection of words. The term independence assumption is also usually adopted, i.e., that the occurrence of each word in a document is independent of the occurrence of other words.

"Bag of Words" / Term Independence Example

If Document 1 contains the text "Mary is quicker than John" and Document 2 contains the text "John is quicker than Mary", then Document 1 & Document 2 are viewed as equivalent.

However, it is unlikely that 30 occurrences of a term in a document truly carries thirty times the significance of a single occurrence of that term. A common modification is to use the logarithm of the term frequency:

$$\begin{aligned} \text{If } tf_{i,d} > 0: &\quad w_{i,d} = 1 + \log(tf_{i,d}) \\ \text{Otherwise:} &\quad w_{i,d} = 0 \end{aligned}$$

4.4.1 Maximum Term Normalisation

We often want to normalise term frequencies because we observe higher frequencies in longer documents merely because longer documents tend to repeat the same words more frequently. Consider a document d' created by concatenating a document d to itself: d' is no more relevant to any query than document d, yet according to the vector space type similarity $\text{sim}(d', q) \geq \text{sim}(d, q) \, \forall \, q$.

The formula for the **maximum term normalisation** of a term i in a document d is usually of the form

$$ntf = a + (1 - a) \frac{tf_{i,d}}{tf\text{max}(d)}$$

where a is a smoothing factor which can be used to dampen the impact of the second term.

Problems with maximum term normalisation include:

- Stopword removal may have effects on the distribution of terms: this normalisation is unstable and may require tuning per collection.
- There is a possibility of outliers with unusually high frequency.
- Those documents with a more even distribution of term frequencies should be treated differently to those with a skewed distribution.

More sophisticated forms of normalisation also exist, which we will explore in the future.

4.4.2 Modern Weighting Schemes

Many, if not all of the developed or learned weighting schemes can be represented in the following format

$$\text{sim}(q,d) = \sum_{t \in q \cap d} \left(ntf(D) \times gw_t(C) \times qw_t(Q) \right)$$

where

- $ntf(D)$ is the normalised term frequency in a document.
- $gw_t(C)$ is the global weight of a term across a collection.
- $qw_t(Q)$ is the query weight of a term in a query Q.

The **Okapi BM25** weighting scheme is a standard benchmark weighting scheme with relatively good performance, although it needs to be tuned per collection:

$$\text{BM25}(Q,D) = \sum_{t \in Q \cap D} \left(\frac{tf_{t,D} \cdot \log\left(\frac{N - df_t 0.5}{df + 0.5}\right) \cdot tf_{t,Q}}{tf_{t,D} + k_1 \cdot \left((1-b) + b \cdot \frac{dl}{dl_{\text{avg}}}\right)} \right)$$

The **Pivoted Normalisation** weighting scheme is also as standard benchmark which needs to be tuned for collection, although it has its issues with normalisation:

$$\text{piv}(Q,D) = \sum_{t \in Q \cap D} \left(\frac{1 + \log\left(1 + \log\left(tf_{t,D}\right)\right)}{(1-s) + s \cdot \frac{dl}{dl_{\text{avg}}}} \right) \times \log\left(\frac{N+1}{df_t}\right) \times tf_{t,Q}$$

The **Axiomatic Approach** to weighting consists of the following constraints:

- **Constraint 1:** adding a query term to a document must always increase the score of that document.
- **Constraint 2:** adding a non-query term to a document must always decrease the score of that document.
- **Constraint 3:** adding successive occurrences of a term to a document must increase the score of that document less with each successive occurrence. Essentially, any term-frequency factor should be sub-linear.
- **Constraint 4:** using a vector length should be a better normalisation factor for retrieval. However, using the vector length will violate one of the existing constraints. Therefore, ensuring that the document length factor is used in a sub-linear function will ensure that repeated appearances of non-query terms are weighted less.

New weighting schemes that adhere to all these constraints outperform the best known benchmarks.