CT4100

Information Retrieval

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

2024–11–07

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.

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

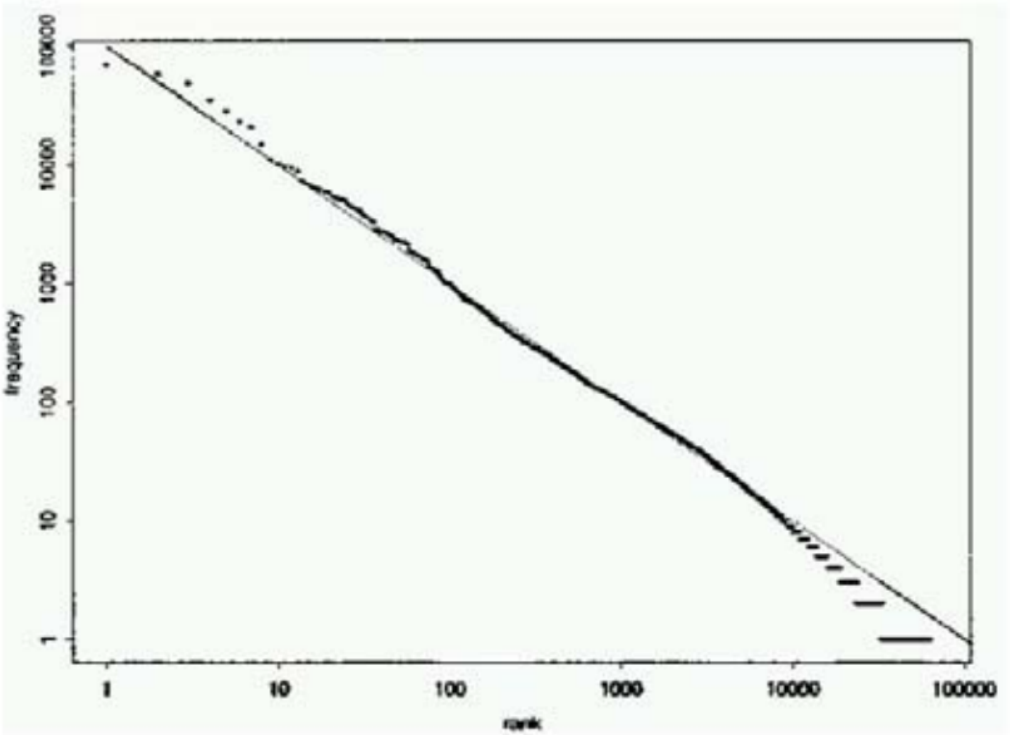

Figure 1: Zipf's Law Modelled on the Brown Corpus

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.

5 Relevance Feedback

We often attempt to improve the performance of an IR system by modifying the user query; the new modified query is then re-submitted to the system. Typically, the user examines the returned list of documents and marks those which are relevant. The new query is usually created via incorporating new terms and re-weighting existing terms. The feedback from the user is used to re-calculate the term weights. Analysis of the document set can either be **local analysis** (on the returned set) or **global analysis** (on the whole document set). This feedback allows for the re-formulation of the query, which has the advantage of shielding the user from the task of query reformulation and from the inner details of the comparison algorithm.

5.1 Feedback in the Vector Space Model

We assume that relevant documents have similarly weighted term vectors. D_r is the set of relevant documents returned, D_n is the set of the non-relevant documents returned, and C_r is the set of relevant documents in the entire collection. If we assume that C_r is known for a query q, then the best vector for a query to distinguish relevant documents from non-relevant documents is

$$\vec{q} = \left(\frac{1}{|C_r|} \sum_{d_j \in C_r} d_j \right) - \left(\frac{1}{N - |C_r|} \sum_{d_j \notin C_r} d_j \right)$$

However, it is impossible to generate this query as we do not know C_r. We can however estimate C_r as we know D_r which is a subset of C_r: the main approach for doing this is the **Rocchio Algorithm:**

$$\overrightarrow{q_{\text{new}}} = \alpha \overrightarrow{q_{\text{orginal}}} + \frac{\beta}{|D_r|} \sum_{d_j \in D_r} d_j - \frac{\gamma}{|D_n|} \sum_{d_j \in D_n} d_j$$

where α, β, & γ are constants which determine the importance of feedback and the relative importance of positive feedback over negative feedback. Variants on this algorithm include:

- **IDE Regular:**

$$\overrightarrow{q_{\text{new}}} = \alpha \overrightarrow{q_{\text{old}}} + \beta \sum_{d_j \in D_r} d_j - \gamma \sum_{d_j \in D_n} d_j$$

- **IDE Dec Hi:** (based on the assumption that positive feedback is more useful than negative feedback)

$$\overrightarrow{q_{\text{new}}} = \alpha \overrightarrow{q_{\text{old}}} + \beta \sum_{d_j \in D_r} d_j - \gamma \text{MAXNR}(d_j)$$

  where $\text{MAXNR}(d_j)$ is the highest ranked non-relevant document.

The use of these feedback mechanisms have shown marked improvement in the precision & recall of IR systems. Salton indicated in early work on the vector space model that these feedback mechanisms result in an average precision of at least 10%.

The precision-recall is re-calculated for the new returned set, often with respect to the returned document set less the set marked by the user.

5.2 Pseudo-Feedback / Blind Feedback

In **local analysis**, the retrieved documents are examined at query time to determine terms for query expansion. We typically develop some form of term-term correlation matrix. To quantify connection between two terms, we expand the query to include terms correlated to the query terms.

5.2.1 Association Clusters

To create an **association cluster**, first create a matrix M; We can create term $\times$ term matrix to represent the level of association between terms. This is usually weighted according to

$$M_{i,j} = \frac{\text{freq}_{i,j}}{\text{freq}_i + \text{freq}_j - \text{freq}_{i,j}}$$

To perform query expansion with local analysis, we can develop an association cluster for each term t_i in the query. For each term $t_i \in q$ choose the i^{th} query term and select the top N values from its row in the term matrix. For a query q, select a cluster for each query term so that $|q|$ clusters are formed. N is usually small to prevent generation of very large queries. We may then either take all terms or just those with the highest summed correlation.

5.2.2 Metric Clusters

Association clusters do not take into account the position of terms within documents: **metric clusters** attempt to overcome this limitation. Let $\text{dis}(t_i, t_j)$ be the distance between two terms t_i & t_j in the same document. If t_i & t_j are in different documents, then $\text{dis}(t_i, t_j) = \text{inf}$. We can define the term-term correlation matrix by the following equation, and we can define clusters as before:

$$M_{i,j} = \sum_{t_i, t_j \in D_i} \frac{1}{\text{dis}(t_i, t_j)}$$

5.2.3 Scalar Clusters

Scalar clusters are based on comparing sets of words: if two terms have similar neighbourhoods then there is a high correlation between terms. Similarity can be based on comparing the two vectors representing the neighbourhoods. This measure can be used to define term-term correlation matrices and the procedure can continue as before.

5.3 Global Analysis

Global analysis is based on analysis of the whole document collection and not just the returned set. A similarity matrix is created with a similar technique to the method used in the vector space comparison. We then index each term by the documents in which the term is contained. It is then possible to calculate the similarity between two terms by taking some measure of the two vectors, e.g. the dot product. To use this to expand a query, we then:

1. Map the query to the document-term space.
2. Calculate the similarity between the query vector and vectors associated with query terms.
3. Rank the vectors $\vec{t_i}$ based on similarity.
4. Choose the top-ranked terms to add to the query.

5.4 Issues with Feedback

The Rocchio & IDE methods can be used in all vector-based approaches. Feedback is an implicit component of many other IR models (e.g., neural networks & probabilistic models). The same approaches with some modifications are used in information filtering. Problems that exist in obtaining user feedback include:

- Users tend not to give a high degree of feedback.
- Users are typically inconsistent with their feedback.
- Explicit user feedback does not have to be strictly binary, we can allow a range of values.
- Implicit feedback can also be used, we can make assumptions that a user found an article useful if:
  - The user reads the article.
  - The user spends a certain amount of time reading the article.
  - The user saves or prints the article.

  However, these metrics are rarely as trustworthy as explicit feedback.

6 Collaborative Filtering

Content filtering is based solely on matching content of items to user's information needs. **Collaborative filtering** collects human judgements and matches people who share the same information needs & tastes. Users share their judgements & opinions. It echoes the "word of mouth" principle. Advantages of collaborative filtering over content filtering include:

- Support for filtering / retrieval of items where contents cannot be easily analysed in an automated manner.
- Ability to filter based on quality / taste.
- Recommend items that do not contain content the user was expecting.

This approach has been successful in a number of domains – mainly in recommending books/music/films and in e-commerce domains, e.g. Amazon, Netflix, Spotify, ebay, etc. It has also been applied to collaborative browsing & searching. In fact, it can be applied whenever we have some notion of "ratings" or "likes" or "relevance" of items for a set of users.

The data in collaborative filtering consists of **users** (a set of user identifiers), **items** (a set of item identifiers), & **ratings by users of items** (numeric values in some pre-defined range). We can usually view this as a user-by-item matrix.

	Item IDs
	0 1 2 3 4 5 6 7 8 9 . . . .
user IDs 0	0 0 0 0 0 0 0 0 0 0 0 0 0 4 3 3 0 0 0 0 3 0 0 0 4 0 0 0 0 0 0 0 0 0
1	0 0 4 3 5 0 3 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2	0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 4 0 4 0
.	0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 4 0 4 3 0 0 0 0
.	0 0 4 0 5 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 3 0 0
.	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 2 4 4 1 0 0 0 0 0
11	0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 2 0 2 0 0 3
12	0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 2 0 0 0 4 0 0 0 1 0 0 4 0 0 0 0 0 0 0
13	0 0 4 3 4 0 0 0 0 0 4 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
.	4 0 0 2 5 0 0 0 4 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
.	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 4

Figure 2: User-By-Item Matrix Example (ratings from 1 to 5; 0 indicates no rating)

With **explicit ratings**, the user usually provides a single numeric value, although the user maybe unwilling to supply many explicit ratings. **Universal queries** are when a gauge set of items is presented to the user for rating. Choosing a good gauge set is an open question. **User-selected queries** are when the user chooses which items to rate (often leaving a sparse ratings matrix with many null values).

Implicit ratings are when the user's recommendation is obtained from purchase records, web logs, time spent reading an item, etc. This implicit rating is usually mapped to some numeric scale.

For **user-user recommendation** approaches, there are three general steps:

1. **Calculate user correlation:** Find how *similar* each user is to every other user.
2. **Select neighbourhood:** form groups or *neighbourhoods* of users who are similar.
3. **Generate prediction:** in each group, *make recommendations* based on what other users in the group have rated.

6.1 Step 1: Calculate User Correlation

Some approaches for finding how similar each user is to every other user include:

- Pearson correlation.

- Constrained Pearson correlation.
- The Spearman rank correlation.
- Vector similarity.

6.1.1 Pearson Correlation

Pearson correlation is when a weighted average of deviations from the neighbour's mean is calculated.

$$w_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \overline{r}_a) \times (r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i=1}^{m}(r_{u,i} - \overline{r}_u)^2} \times \sqrt{\sum_{i=1}^{m}(r_{a,i} - \overline{r}_a)^2}}$$

where for m items:

- $r_{a,i}$ is the rating of a user a for an item i.
- r_a is the average rating given by a user a.
- $r_{u,i}$ is the rating of user u for item i.
- r_u is the average rating given by user u.

6.1.2 Vector Similarity

Vector similarity uses the cosine measure between the user vectors (where users are represented by a vector of ratings for items in the data set) to calculate correlation.

6.2 Step 2: Select Neighbourhood

Some approaches for forming groups or **neighbourhoods** of users who are similar include:

- **Correlation thresholding:** all neighbours with absolute correlations greater than a specified threshold are selected, say 0.7 if correlations in range 0 to 1.
- **Best-n correlations:** the best n correlates are chosen.

A large neighbourhood can result in low-precision results, while a small neighbourhood can result in few or now predictions.

6.3 Step 3: Generate Predictions

For some user (the active user) in a group, make recommendations based on what other users inthe group have rated which the active user has not rated. Approaches for doing so include:

- **Compute the weighted average** of the user rating using the correlations as the weights. This weighted average approach makes an assumption that all users rate items with approximately the same distribution.
- **Compute the weighted mean** of all neighbours' ratings. Rather than take the explicit numeric value of a rating, a rating's strength is interpreted as its distance from a neighbour's mean rating. This approach attempts to account for the lack of uniformity in ratings.

  $$P_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n}(r_{u,i-\bar{r}_u}) \times w_{a,u}}{\sum_{u=1}^{n} w_{a,u}}$$

  where for n neighbours:

  - $\bar{r}_a$ is the average rating given by active user a.
  - $r_{u,i}$ is the rating of user u for item i.
  - $w_{a,u}$ is the similarity between user u and a.

Note that the Pearson Correlation formula does not explicitly take into account the number of co-rated items by users. Thus it is possible to get a high correlation value based on only one co-rated item. Often, the Pearson Correlation formula is adjusted to take this into account.

6.4 Experimental Approach for Testing

A known collection of ratings by users over a range of items is decomposed into two disjoint subsets. The first set (usually the larger) is used to generate recommendations for items corresponding to those in the smaller set. These recommendations are then compared to the actual ratings in the second subset. The accuracy & coverage of a system can thus be ascertained.

6.4.1 Metrics

The main metrics used to test the predictions produced are:

- **Coverage:** a measure of the ability of the system to provide a recommendation on a given item.
- **Accuracy:** a measure of the correctness of the recommendations generated by the system.

Statistical accuracy metrics are usually calculated by comparing the ratings generated by the system to user-provided ratings. The accuracy is usually presented as the mean absolute error (**MAE**) between ratings & predictions.

Typically, the value of the rating is not that important: it is more important to know if the rating is a useful or a non-useful rating. **Decision support accuracy metrics** measure whether the recommendation is actually useful to the user. Many other approaches also exist, including:

- Machine learning approaches.
  - Bayesian models.
  - Clustering models.
- Models of how people rate items.
- Data mining approaches.
- Hybrid models which combine collaborative filtering with content filtering.
- Graph decomposition approaches.

6.5 Collaborative Filtering Issues

- **Sparsity of Matrix:** in a typical domain, there would be many users & many items but any user would only have rated a small fraction of all items in the dataset. Using a technique such as **Singular Value Decomposition (SVD)**, the data space can be reduced, and due to this reduction a correlation may be found between similar users who do not have overlapping ratings in the original matrix of ratings.
- **Size of Matrix:** in general, the matrix is very large, which can affect computational efficiency. SVD has been used to improve scalability by dimensionality reduction.
- **Noise in Matrix:** we need to consider how would a user's ratings change for items and how to model time dependencies; are all ratings honest & reliable?
- **Size of Neighbourhood:** while the size of the neighbourhood affects predictions, there is no way to know the "right" size. Need to consider whether visualisation of the would neighbourhood help, whether summarisation of the main themes/feature of neighbourhoods help.
- **How to Gather Ratings:** new users, new items: perhaps use weighted average of global mean & users or items. What if the user is not similar to others?

6.6 Combining Content & Collaborative Filtering

For most items rated in a collaborative filtering domain, content information is also available:

- Books: author, genre, plot summary, language, etc.
- Music: artist, genre, sound samples, etc.
- Films: director, genre, actors, year, country, etc.

Traditionally, content is not used in collaborative filtering, although it could be.

Different approaches may suffer from different problems, so can consider combining multiple approaches. We can also view collaborative filtering as a machine learning classification problem: for an item, do we classify it as relevant to a user or not?

Much recent work has been focused on not only giving a recommendation, but also attempting to explain the recommendation to the user. Questions arise in how best to "explain" or visualise the recommendation.

7 Learning in Information Retrieval

Many real-world problems are complex and it is difficult to specify (algorithmically) how to solve many of these problems. Learning techniques are used in many domains to find solutions to problems that may not be obvious or clear to human users. In general, machine learning involves searching a large space of potential hypotheses or potential solutions to find the hypotheses/solution that best *explains* or *fits* a set of data and any prior knowledge, or is the best solution, or the solution that we can say learns if it improves the performance.

Machine learning techniques require a training stage before the learned solution can be used on new previously unseen data. The training stage consists of a data set of examples which can either be:

- **Labelled** (supervised learning).
- **Unlabelled** (unsupervised learning).

An additional data set must also be used to test the hypothesis/solution.

Symbolic knowledge is represented in the form of the symbolic descriptions of the learned concepts, e.g., production rules or concept hierarchies. **Sub-symbolic knowledge** is represented in sub-symbolic form not readable by a user, e.g., in the structure, weights, & biases of the trained network.

7.1 Genetic Algorithms

Genetic algorithms are inspired by the Darwinian theory of evolution: at each step of the algorithm, the best solutions are selected while the weaker solutions are discarded. It uses operators based on crossover & mutation as the basis of the algorithm to sample the space of solutions. The steps of a genetic algorithm are as follows: first, create a random population. Then, while a solution has not been found:

1. Calculate the fitness of each individual.
2. Select the population for reproduction:
    i. Perform crossover.
    ii. Perform mutation.
3. Repeat.

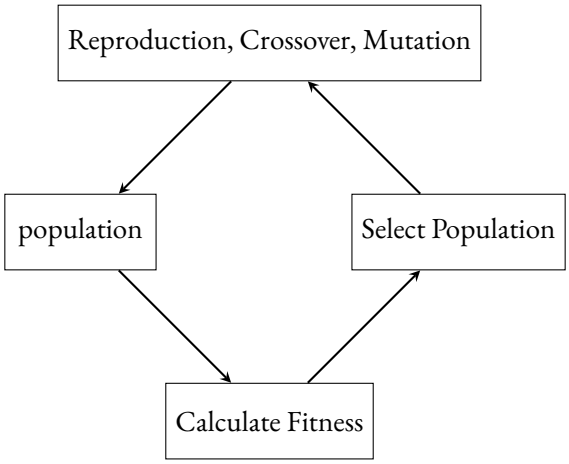

Figure 3: Genetic Algorithm Steps

Traditionally, solutions are represented in binary. A **phenotype** is the decoding or manifestation of a **genotype** which is the encoding or representation of a phenotype. We need an evaluation function which will discriminate between better and worse solutions.

Crossover Examples

Example of one-point crossover: `11001011` and `11011111` gives `11001111` and `11011011`.

Example of n-point crossover: `110110110` and `0001001000` gives `1101001100` and `0001101001`.

Mutation occurs in the genetic algorithm at a much lower rate than crossover. It is important to add some diversity to the population in the hope that new better solutions are discovered and therefore it aids in the evolution of the population.

Mutation Example

Example of mutation: `11001001` $\rightarrow$ `10001001`.

There are two types of selection:

- **Roulette wheel selection:** each sector in the wheel is proportional to an individual's fitness. Select n individuals by means of n roulette turns. Each individual is drawn independently.
- **Tournament selection:** a number of individuals are selected at random with replacement from the population. The individual with the best score is selected. This is repeated n times.

Issues with genetic algorithms include:

- Choice of representation for encoding individuals.
- Definition of fitness function.
- Definition of selection scheme.
- Definition of suitable genetic operators.
- Setting of parameters:
  - Size of population.
  - Number of generations.
  - Probability of crossover.

– Probability of mutation.

Case Study 1: Application of Genetic Algorithms to IR

The effectiveness of an IR system is dependent on the quality of the weights assigned to terms in documents. We have seen heuristic-based approaches & their effectiveness and we've seen axiomatic approaches that could be considered.

Why not learn the weights? We have a definition of relevant & non-relevant documents; we can use MAP or precision@k as fitness. Each genotype can be a set of vectors of length N (the size of the lexicon). Set all rates randomly initially. Run the system with a set of queries to obtain fitness; select good chromosomes; crossover; mutate. Effectively searching the landscape for weights to give a good ranking.

7.2 Genetic Programming

Genetic programming applies the approach of the genetic algorithm to the space of possible computer programs. "Virtually all problems in artificial intelligence, machine learning, adaptive systems, & automated learning can be recast as a search for a computer program. Genetic programming provides a way to successfully conduct the search for a computer program in the space of computer programs." – Koza.

A random population of solutions is created which are modelled in a tree structure with operators as internal nodes and operands as leaf nodes.

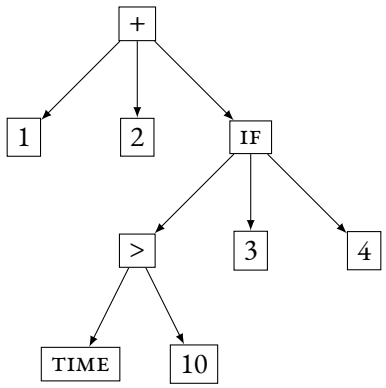

Figure 4: `(+ 1 2 (IF (> TIME 10) 3 4))`

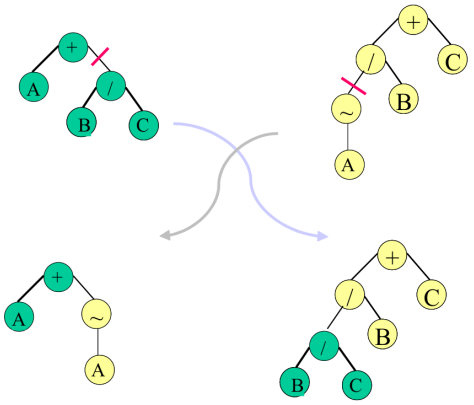

Figure 5: Crossover Example

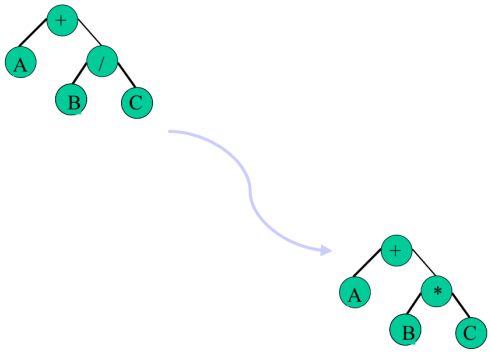

Figure 6: Mutation Example

The genetic programming flow is as follows:

1. Trees are (usually) created at random.
2. Evaluate how each tree performs in its environment (using a fitness function).
3. Selection occurs based on fitness (tournament selection).
4. Crossover of selected solutions to create new individuals.
5. Repeat until population is replaced.
6. Repeat for N generations.

7.2.1 Anatomy of a Term-Weighting Scheme

Typical components of term weighting schemes include:

- Term frequency aspect.
- "Inverse document" score.
- Normalisation factor.

The search space should be decomposed accordingly.

7.2.2 Why Separate Learning into Stages?

The search space using primitive measures & functions is extremely large; reducing the search space is advantageous as efficiency is increased. It eases the analysis of the solutions produced at each stage. Comparisons to existing benchmarks at each of these stages can be used to determine if the GP is finding novel solutions or variations on existing solutions. It can then be identified from where any improvement in performance is coming.

7.2.3 Learning Each of the Three Parts in Turn

1. Learn a term-discrimination scheme (i.e., some type of idf) using primitive global measures.
   - 8 terminals & 8 functions.
   - $T = \{df, cf, N, V, C, 1, 10, 0.5\}$.
   - $F = \{+, \times, \div, -, \text{square}(), \text{sqrt}(), \ln(), \exp()\}$.
2. Use this global measure and learn a term-frequency aspect.
   - 4 terminals & 8 functions.
   - $T = \{tf, 1, 10, 0.4\}$.

- $F = \{+, \times, \div, -, \text{square}(), \text{sqrt}(), \ln(), \exp()\}$.

3. Finally, learn a normalisation scheme.
   - 6 terminals & 8 functions.
   - $T = \{\text{dl}, \text{dl}_{\text{avg}}, \text{dl}_{\text{dev}}, 1, 10, 0.5\}$.
   - $F = \{+, \times, \div, -, \text{square}(), \text{sqrt}(), \ln(), \exp()\}$.

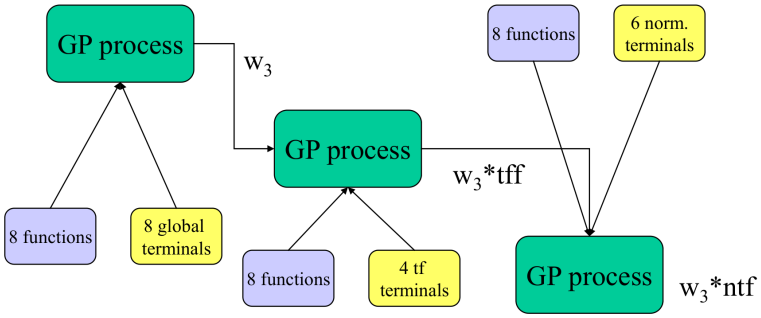

Figure 7: Learning Each of the Three Stages in Turn

7.2.4 Details of the Learning Approach

- 7 global functions were developed on 32,000 OHSUMED documents.
  - All validated on a larger unseen collection and the best function taken.
  - Random population of 100 for 50 generations.
  - The fitness function used was MAP.
- 7 tf functions were developed on 32,000 LATIMES documents.
  - All validated on a larger unseen collection and the best function taken.
  - Random population of 200 for 25 generations.
  - The fitness function used was MAP.
- 7 normalisation functions were developed 3 $\times$ 10,000 LATIMES documents.
  - All validated on a larger unseen collection and the best function taken.
  - Random population of 200 for 25 generations.
  - Fitness function used was average MAP over the 3 collections.

7.2.5 Analysis

The global function w_3 always produces a positive number:

$$w_3 = \sqrt{\frac{cf_t^3 \cdot N}{df_t^4}}$$

Case Study 1: Application of Genetic Programming to IR

Evolutionary computing approaches include:

- Evolutionary strategies.
- Genetic algorithms.
- Genetic programming.

Why genetic programming for IR?

- Produces a symbolic representation of a solution which is useful for further analysis.
- Using training data, MAP can be directly optimised (i.e., used as the fitness function).
- Solutions produced are often generalisable as solution length (size) can be controlled.

Empirical evaluation shows that the evolved scheme outperforms a tuned pivot normalisation scheme and a tuned BM25 scheme. The evolved scheme is also non-parametric. The use of primitive atomic measures and basic function types is crucial in allowing the shape of term-weighting functions to evolve.

7.3 Neural Networks

Previously, we reviewed the notion of learning in IR and looked at the application of the evolutionary computation approach as a search for solutions in information retrieval. The advantages were that we had solutions that could be analysed. However, the usefulness of the solution found is dependent on the usefulness of the primitive features chosen to extract from queries and the documents collection.

The dominant learning approach in recent years has been the **neural** approach. The neural approach can be seen being applied directly to:

- The information retrieval tasks itself.
- To other related problems/areas that can feed into the IR process.
- Related problems in IR (e.g., query suggestions).

Approaches in the domain have been both supervised & unsupervised. One of the first approaches to adopt a neural network model can be traced back to the 1980s, with an effectively three-layer network consisting of documents, terms, & query. A spreading activation method was used, where query nodes are highlights and propagate to terms which in turn highlight certain documents.

Case Study: Self-Organising Maps (Kohonen)

Self-organising maps are an example of unsupervised learning. Documents are mapped to 2D space, and dense areas indicate clusters hierarchically. In the Kohonen approach, documents are mapped to 2D space and each region is characterised / represented by terms.

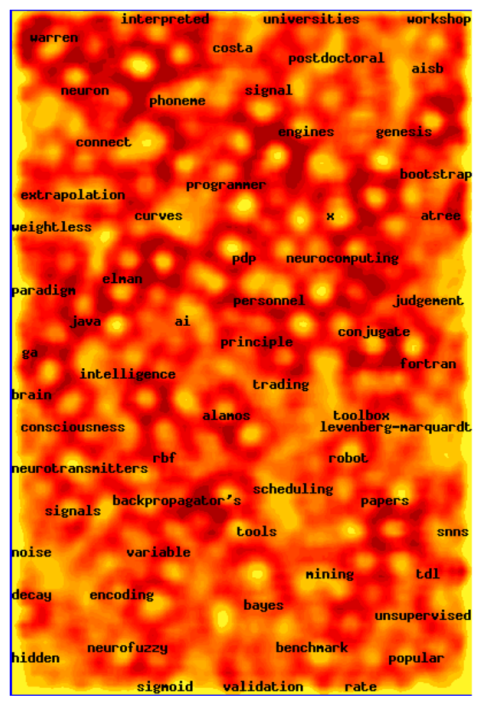

Figure 8: A SOM created over a collection of AI-related documents

Users can traverse the collection by clicking on an area of the map that is of interest.

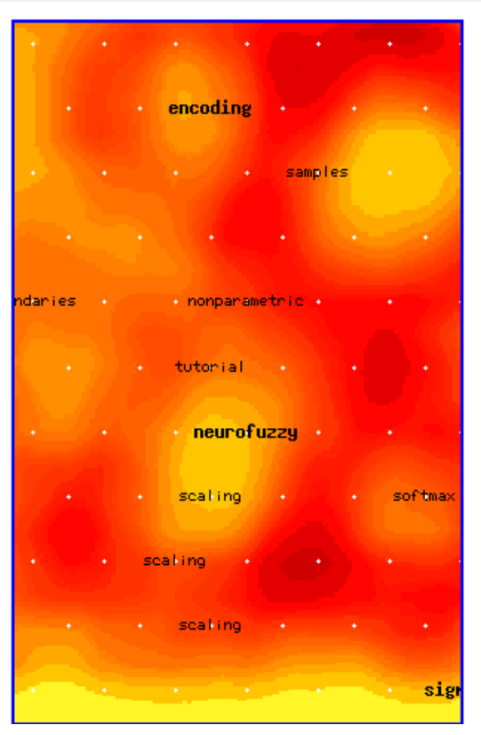

Figure 9: Finally, the user arrives at a list of papers/articles that have been clustered together

Kohonen self-organising maps represent a sub-symbolic, neural approach to clustering. The algorithm takes a set of n-dimensional vectors and attempts to map them onto a two-dimensional grid. The grid comprises of a set of nodes, each of which is assigned an n-dimensional vector. These vectors contain randomly assigned weights.

The algorithm is as follows:

1. Select an input vector randomly.
2. Identify the grid node which is closest to the input vector: the "winning node".
3. Adjust the weights on the winning node so that it is closer to the input vector.
4. Adjust the weights on nodes near to the winning node so that they are closer to the winning node.

Note:

- The rate of modification of weights decreases over time.
- The size of the neighbourhood affected (near the winning node) decreases over time.
- The resulting clustering of tuples maintains the distance relationship between the input data.

There has been huge interest in the application of NN models in IR in recent years as there have been several breakthroughs due to the use of neural networks with multiple layers (so-called "deep architectures") and the availability of large datasets & computing power. Proposed neural models learn representations of language from raw text that can bridge the gap between query & document vocabulary. Neural IR is the application of shallow or deep NN to IR tasks.

Neural models for IR use vector representations of text and usually contain a large number of parameters that need to be tuned. ML models with a large set of parameters benefit from large quantities of training data. Unlike traditional "learning to rank" approaches that train over a set of hand-crafted features, more recent NN accept the raw text as input. Some main steps in classical IR include:

- Generating a representation of the user's query.
- Generating a representation of the documents that captures the "content" of the document.
- Generating a "similarity" score (comparison).

All neural approaches can be classified as to whether they affect the representation of the query, the document, or the comparison. By inspecting only the query terms, the IR model ignores all evidence of the context provided in the rest of the document: only occurrences of a word are included, and not other terms that capture the same meaning or the same topic. Traditional IR models have also used dense vector representations of terms & documents. Many neural representations have commonalities with these traditional approaches.

7.4 Query Representation

Types of vector representations include:

- One-Hot representations: akin to what we have seen thus far, in that each term is represented by one value.
- Distributed representation: typically a real-valued vector which attempts to better capture the meaning of the terms.

NNs are often used to learn this "embedding" or representation of terms.

The **distributional hypothesis** states that terms that occur in similar contexts tend to be semantically similar. An **embedding** is a representation of items in a new space such that the properties of, and the relationships between the items are preserved from the original representation. There are many algorithms for this, e.g., *word2vec*. Representations are generated for queries & for documents. We compare the query & document in this embedding space: documents & queries that are similar should be similar in this embedding space.

Text & language is typically represented as a sequence; for analysing questions & sentences, we need to learn or model

these sequences. In a **recurrent neural network**, a neuron's output is a function of the current state of the neuron & the input vector. They are very successful in capturing / learning sequential relationships. A large set of architectures are used with different topologies. Convolutional networks (most often associated with images) are also used to learn the relationships between terms. Sequential processing has been used in query understanding, retrieval, expansion, etc.

In summary, neural approaches are powerful, typically more computationally expensive than traditional approaches, have good performance, but have issues with explainability.

8 Clustering

8.1 Introduction

Document clustering is the process of grouping a set of documents into clusters of similar documents. Documents within a cluster should be similar, while documents from different clusters should be dissimilar. Clustering is the most common form of **unsupervised learning**, i.e., there is no labelled or annotated data.

8.1.1 Classification vs Clustering

Classification is a supervised learning algorithm in which classes are human-defined and part of the input to the algorithm. Clustering is an unsupervised learning algorithm in which clusters are inferred from the data without human input. However, there are many ways of influencing the outcome of clustering: number of clusters, similarity measure, representation of documents, etc.

8.2 Clustering in IR

The **cluster hypothesis** states that documents in the same cluster behave similarly with respect to relevance information needs. All applications of clustering in IR are based (directly or indirectly) on the cluster hypothesis. Van Rijsbergen's original wording of the cluster hypothesis was "closely related documents tend to be relevant to the same requests".

Applications of clustering include:

- Search result clustering: search results are clustered to provide more effective information presentation to the user.
- Scatter-gather clustering: (subsets of) the collection are clustered to provide an alternative user interface wherein the user can search without typing.
- Collection clustering: the collection is clustered for effective information presentation for exploratory browsing.
- Cluster-based retrieval: the collection is clustered to provide higher efficiency & faster search.

8.2.1 Clustering for Improving Recall

To improve search recall:

1. Cluster documents in collection *a priori*.
2. When a query matches a document d, also return other documents in the cluster that contains d.

The hope is that if we do this, the query "car" will also return documents containing "automobile", as the clustering algorithm groups together documents containing "car" with those containing "automobile". Both types of documents will contain words like "parts", "dealer", "mercedes", "road trip".

8.2.2 Desiderata for Clustering

- The general goal is to put related documents in the same cluster and to put unrelated documents in different clusters.
- The number of clusters should be appropriate for the data set we are gathering
- Secondary goals in clustering include:
  - Avoid very small & very large clusters.
  - Define clusters that are easy to explain to the user.

8.2.3 Flat vs Hierarchical Clustering

Flat algorithms usually start with a random (partial) partitioning of documents into groups and are refined iteratively. The main example of this is k-means. Flat algorithms compute a partition of n documents into a set of k clusters. Given a set of documents and the number k, the goal is to find a partition into k clusters that optimises the chosen partitioning criterion. Global optimisation may be achieved by exhaustively enumerating partitions and picking the optimal one, however this is not tractable. The k-means algorithm is an effective heuristic method.

Hierarchical algorithms create a hierarchy: either top down, agglomerative, or top-down, divisive.

8.2.4 Hard vs Soft Clustering

In **hard clustering**, each document belongs to exactly one cluster. This is more common, and easier to do.

Soft clustering: a document can belong to more than one cluster; this makes sense for applications like creating browsable hierarchies. For example, you may want to put the word "sneakers" in two clusters: sports apparel & shoes.

8.3 k-Means

k-means is perhaps the best-known clustering algorithm, as it is simple and works well in many cases. It is used as a default / baseline for clustering documents. Document representation in clustering are typically done using the vector space model, with the relatedness between vectors being measured by Euclidean distance.

Each cluster in k-means is defined by a **centroid**, which is defined as:

$$\vec{\mu}(\omega) = \frac{1}{|\omega|} \sum_{\vec{x} \in \omega} \vec{x}$$

where ω defines a cluster.

The partitioning criterion is to minimises the average squared difference from the centroid. We try to find the minimum average squared difference by iterating two steps:

- **Reassignment:** assign each vector to its closest centroid.
- **Recomputation:** recompute each centroid as the average of the vectors that were assigned to it in reassignmnet.

Algorithm 1 k-means$(\{\vec{x}_1, \ldots, \vec{x}_N\}, k)$

```
(s_1, s_2, ..., s_K) ← SelectRandomSeeds({x_1, ..., x_N}, k)
for k ← 1 to k do
    μ_k ← s_k
end for
while stopping criterion has not been met do
    for k ← 1 to k do
        ω_k ← {}
    end for
    for n ← 1 to N do
        j ← arg min_j ||μ_j − x_n||
        ω_j ← ω_j ∪ {x_n}                          ▷ reassignment of vectors
    end for
    for k ← 1 to k do
        μ_k ← (1/|ω_k|) Σ_{x∈ω_k} x                ▷ recomputation of centroids
    end for
end while
return {μ_1, ..., μ_k}
```

8.3.1 Proof that k-Means is Guaranteed to Converge

1. RSS is the sum of all squared distances between the document vector and the closest centroid.
2. RSS decreases during each reassignment step because each vector is moved to closer a centroid.
3. RSS decreases during each recomputation step.
4. There is only a finite number of clusterings, thus we must reach a fixed point.

However, we don't know how long convergence will take. If we don't care about a few documents switching back and forth, then convergence is usually fast (around 10-20 iterations). However, complete convergence can take many more iterations.

The great weakness of k-means is that **convergence does not mean that we converge to the optimal clustering**. If we start with a bad set of seeds, the resulting clustering can be poor.

8.3.2 Initialisation of k-means

Random seed selection is just one of many ways k-means can be initialised. Random seed clustering is not very robust: it's very easy to get a sub-optimal clustering. Better ways of computing initial centroids include:

- Select seeds not randomly, but using some heuristic, e.g., filter out outliers or find a set of seeds that has "good coverage" of the document space.
- Use hierarchical clustering to find good seeds.
- Select i (e.g., $i = 10$ different random sets of seeds), and do a k-means clustering for each before selecting the clustering with the lowest RSS.

8.4 Evaluation

Internal criteria for a clustering, e.g. RSS in k-means often do not evaluate the actual utility of a clustering in the application. An alternative to internal criteria is external criteria, i.e. to evaluate with respect to a human-defined classification.

8.4.1 External Criteria for Clustering Quality

External criteria for clustering quality are based on the ideal "gold standard" dataset, where the goal is that clustering should reproduce the classes in the gold standard.

Purity measures how well we were able to reproduce the classes:

$$\text{purity}(\Omega, C) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

where $\Omega = \{\omega_1, \omega_2, \ldots, \omega_k\}$ is the set of clusters and $C = \{c_1, c_2, \ldots, c_j\}$ is the set of classes. For each cluster ω_k, find the class c_j with the most members n_{kj} in ω_k. Sum all n_{kj} and divide by the total number of points.

The **Rand Index** is defined as:

$$\text{RI} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

It is based on a 2×2 contingency table of all pairs of documents: $\text{TP} + \text{FP} + \text{FN} + \text{TN}$ is the total number of pairs. There are $\binom{N}{2}$ pairs for N documents. Each pair is either positive or negative as the clustering either puts the documents in the same or in different clusters. Each pair is also either "true" (correct) or "false" (incorrect), i.e., the clustering decision is either correct or incorrect.

	same cluster	**different clusters**
same class	TP	FN
different classes	FP	TN

Table 2: 2×2 contingency table of all pairs of documents

8.4.2 How Many Clusters?

The number of clusters k is given in many applications.