Relevance Feedback (and Pseudo Relevance Feedback)

COMP3009J: Information Retrieval

Dr. David Lillis (david.lillis@ucd.ie)

UCD School of Computer Science
Beijing Dublin International College

Reference: Section 9.1, *An Introduction to Information Retrieval*, Manning, Raghavan & Schütze

Relevance Feedback

- **Relevance Feedback** is when the user is involved in the retrieval process to improve the final result set.
- Specifically the user gives feedback on the **relevance** of documents in the initial set of results, and this is used to re-process their query and provide an updated set of results.
- Although this is not typically used in most modern IR systems, the process is important because **pseudo relevance feedback** is common, which is a simulation of relevance feedback without the user being directly involved.

Relevance Feedback Process

- Typical process:
  1. User **provides a query** (usually short and simple)
  2. The system returns an **initial set of results**.
  3. The user marks some **relevant** and **nonrelevant** documents (in some situations the user only marks relevant results).
  4. The system computes a **better representation of the information need** based on this feedback.
  5. The system returns a **revised set of results**.
  6. Possibly the above process is repeated.

Rocchio Algorithm: Background

- Probably the best-known algorithm for relevance feedback is the **Rocchio Algorithm**.
- Basic Idea:
  - Based on the idea that documents and queries are **represented as vectors** (e.g. using TF-IDF, although any vector representation will work).
  - Wants to **find a query vector** that maximises similarity with relevant documents and minimises similarity with nonrelevant documents.

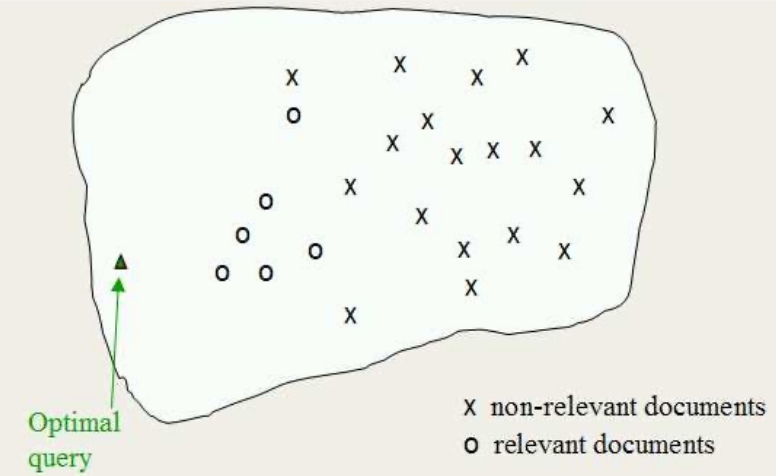
Rocchio Algorithm: Background

- Formally the aim is to find an optimal query vector $\vec{q}_{opt}$.
- The optimal query vector is the query vector with the largest difference between its similarity with the set of relevant documents and its similarity with the set of nonrelevant documents.

$$\vec{q}_{opt} = \arg\max_{\vec{q}} [sim(\vec{q}, C_r) - sim(\vec{q}, C_{nr})]$$

- where:
  - $sim(\ldots)$ is a measure of similarity (e.g. cosine similarity)
  - C_r is the set of relevant documents
  - C_{nr} is the set of nonrelevant documents

Rocchio Algorithm: Background

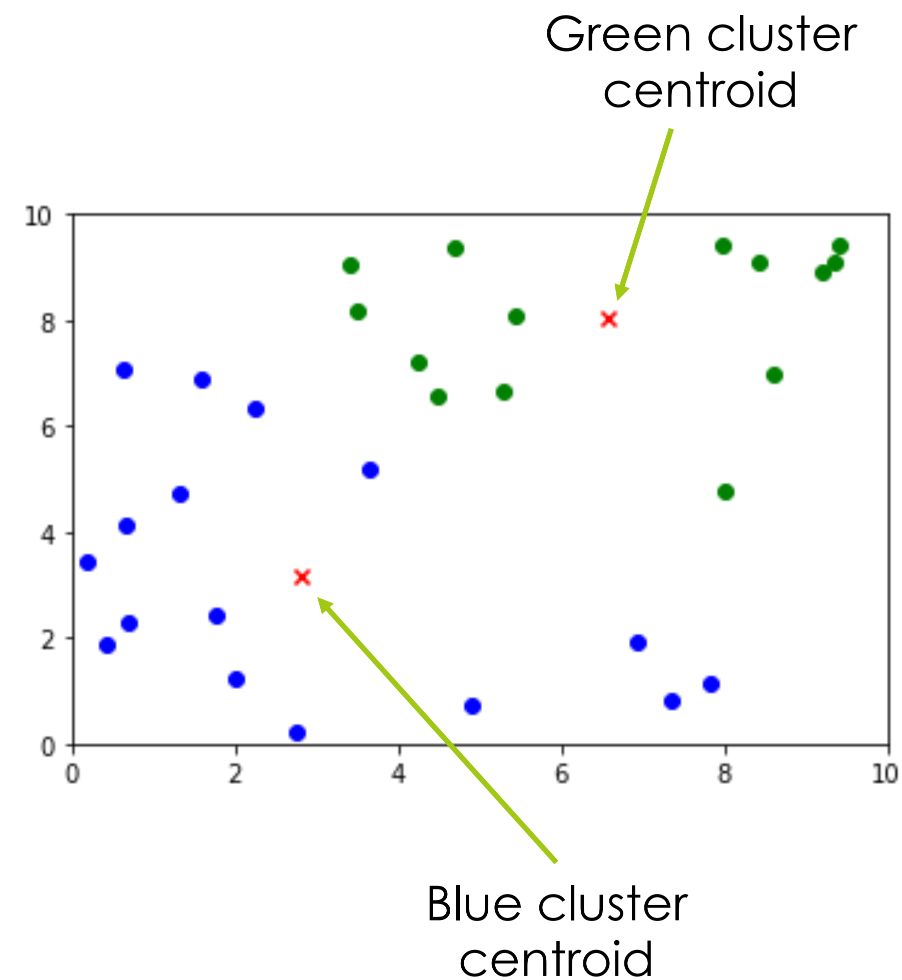


- This illustrates the ideal situation.
- Of course, the real vector space has far more than 2 dimensions: this is only an illustration.

Centroids

To further examine how the similarity between a query vector and a set of documents can be calculated, we need to consider the idea of a **centroid** of a set of vectors.

The centroid is the centre point of a set of vectors. It is itself a vector.



$$Centroid(C) = \frac{1}{|C|} \sum_{\vec{d}_j \in C} \vec{d}_j$$

Rocchio Algorithm: Theory With complete knowledge

- Using cosine similarity, the formula for calculating the optimal query vector is:

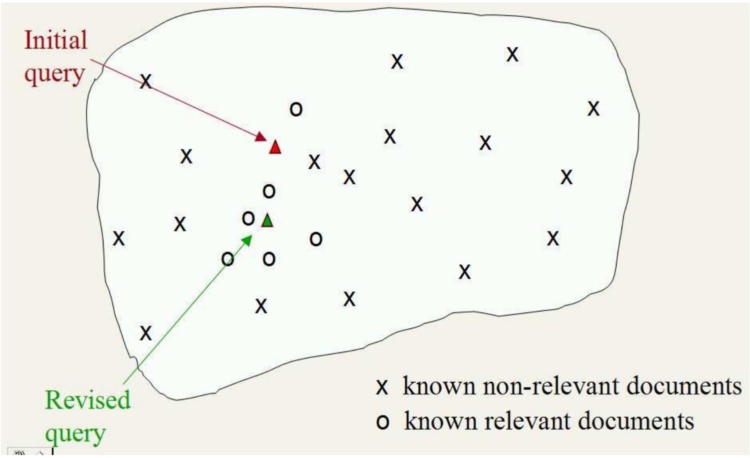
$$\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\vec{d}_j \in C_r} \vec{d}_j - \frac{1}{|C_{nr}|} \sum_{\vec{d}_j \in C_{nr}} \vec{d}_j$$

- Informally, this is the vector that is obtained by subtracting the centroid of the nonrelevant set from the centroid of the relevant set.
- **But:** In practice we don't yet know the relevant and nonrelevant sets!

Rocchio Algorithm in practice

In practice, in relevance feedback situations, we have access to a subset of the relevant and nonrelevant sets, which have been identified by the user.

The Rocchio algorithm takes this information and computes a **modified query vector** (or “revised query vector”), which is probably not be the optimal query vector, and re-runs retrieval.



Rocchio Algorithm: Formula

- The formula to calculate the **modified query vector** ($\vec{q}_m$) is as follows:

$$\vec{q}_m = \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r}\vec{d}_j - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}}\vec{d}_j$$

- Here:
- $\vec{q}_0$ is the original query provided by the user.
- D_r and D_{nr} are the documents that have been identified by the user as being relevant and nonrelevant respectively.
- α, β and γ are weights that can affect the behaviour of the formula.

Rocchio Algorithm: Weights

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

- From this formula, what is the effect of changing the weights α, β and γ?
  - They control the balance between trusting the judged sets of relevant and nonrelevant documents and trusting the original query.
  - With many judged documents, β and γ might be high.
  - The modified query moves the original query some distance away from the centroid of the nonrelevant documents (γ) and some distance towards the centroid of the relevant documents (β).

Rocchio Algorithm in Practice

- In practice, relevance feedback is very helpful to increase recall in situations where that is important.
  - This is also the most likely situation where a user is happy to provide feedback.
  - Positive feedback (i.e. identifying relevant documents) is more useful than negative feedback (i.e. identifying non-relevant documents), so in most systems $\gamma < \beta$.
  - Reasonable values might be $\alpha = 1$, $\beta = 0.75$ and $\gamma = 0.15$.
    - Some systems only support positive feedback, which is effectively setting $\gamma = 0$.

Relevance Feedback Assumptions

- The user must have enough knowledge to make an **initial query that is close** to the relevant documents.
  - Without this, they will not be shown any relevant documents that they can give feedback on.
- Some problems that relevance feedback cannot solve by itself:
  - **Misspellings**: if the user's query does not spell terms correctly.
  - **Cross-language IR**: where documents in different languages will not be close in the vector space (although this depends on the vector representation).
  - **Vocabulary Mismatch**: where the user uses a different word/phrase to describe something (e.g. "laptop" versus "notebook computer").

Relevance Feedback Assumptions

- Relevant documents are assumed to be **close to each other** in a cluster.
- **But** sometimes relevant documents can be in several clusters:
  - Documents might use different vocabulary (e.g. astronaut vs. cosmonaut).
  - A query that combines two very different things (e.g. Football players who studied at UCD).
  - General concepts that combine many more specific subjects (e.g. ”felines” includes both wild animals and domestic pets).

Pseudo Relevance Feedback

- In practice, users generally do not wish to provide feedback (known as **explicit** feedback).
- Instead, **pseudo relevance feedback** (also called **blind relevance feedback**) can be used: assume the top *k* ranked documents are relevant and then apply a relevance feedback algorithm.
  - Quite successful in practice, although it can have problems in some situations.

Indirect Relevance Feedback

- If users do not provide explicit feedback by actively marking documents as relevant or nonrelevant, other sources of evidence can be gathered to get **implicit** feedback. For example:
  - Which documents does the user choose to click on?
  - How long did the user spend viewing a document?
  - How long did the user spend browsing the results?

Summary

- **Relevance feedback** allows the user to participate in retrieval by informing the system of results that are relevant and/or ones that are non-relevant.
  - **Example:** Rocchio algorithm based on vector representation.
- However, this is **not popular in practice**, so other ways are used to simulate feedback or get some feedback from other sources:
  - **Pseudo Relevance Feedback** (or “blind relevance feedback”: assume top *k* documents are relevant).
  - **Implicit Relevance Feedback** (evidence from other sources: clicks and user behaviour).