University of Sheffield

Medical study identification for the ScHARRHUD database using text mining techniques

Supervisor: Mark Stevenson

William Goldsworthy

Department of Computer Science

May 1, 2018

Declaration

Name: William Goldsworthy

Signature: William Goldsworthy

Date: 01.5.18

Abstract

There is currently a large and growing number of published medical studies which makes the identification of relevant studies for systematic reviews increasingly difficult and time consuming. Experts are forced to manually sift through thousands of irrelevant studies in order to identify those that are relevant and useful for their review. ScHARRHUD is an innovative database containing bibliographic details of medical studies which report on health state utility values (HSUVs). This project aims to use text mining techniques to identify studies similar to those already included in the database and consequently improve the quality of the collection. This project will produce a list of relevant studies that can be added to the ScHARRHUD database.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

In the medical research space, there are approximately two thousand papers published daily adding to an already immense collection [2]. Given the rate of increase, the task of screening studies to identify which are relevant to a particular area has become time consuming. Whilst PubMed, an online database of medical papers, has dramatically improved the availability of and access to medical papers, it has also resulted in an increase to the number of papers to be manually screened. The use of text mining techniques is a proposed solution to reduce both time and resources required. Text mining can be used to automate part of the screening process and to rank documents in order of similarity to a particular query so that the most relevant can be manually screened first.

1.2 Aims and objectives

The aim of this project is to use text mining techniques to identify relevant documents for use in the ScHARRHUD database. ScHARR, the School of Health and Related Research, is part of the University of Sheffield and 'tackles some of the world's biggest health challenges to improve the health and care of people in the UK and around the world'. This project is focused around ScHARRHUD, 'an innovative database aimed at improving access to health utilities evidence'. More specifically, the database holds the bibliographic details of studies reporting health state utility values (HSUVs). Finding similar studies which include HSUVs is the main objective of this project aiming to reduce the workload of ScHARR employees and to further improve the quality of the ScHARRHUD database. In particular, the project aims to produce a ranked list of studies in order of their similarity to the collection already held in the ScHARRHUD database. There is future scope to use active learning to generate an improved set of results. Active learning is a form of machine learning that can be used to repeatedly generate improved sets of data with additional user input based on the validity of the previously generated set.

1.3 System produced for this project

This project produces a system to aid ScHARR in finding relevant documents for their database, making use of a range of text mining techniques. Initially the current set of studies in the ScHARRHUD database are pre-processed, using stemming and the removal of stop words. The system uses the tf.idf vectorization to convert both the current collection and the search collection of documents into vectors in order for the cosine similarity to be calculated. From this, a ranked list of new documents and their similarity to the current set is produced and returned as the results of the system. Whilst there is the potential to turn the scripts into a reusable tool, more user friendly than a command line script, this project is essentially focused around the results produced by the scripts rather than their reusability. The project also implements some active learning in order to further improve the performance of the system.

1.4 Overview of the report

This report will cover the currently available literature relating to medical study screening and the application of text mining in this area. Chapter 2 details the background of the text mining techniques and the process of evaluation. There is a particular focus on the current use of text mining in medical study screening and its effectiveness at both reducing overall workload and producing quality results. Finally, there is a discussion of active learning, relevance feedback and its use within the area of medical study identification.

Chapter 3 describes the requirements of the project and will break the project down into individual components. The requirements from ScHARR will be described as well as how the system will be tested and evaluated.

Chapter 4 explains the design of the system and why this particular approach was taken.

The methods for solving the issues arising from the project are described in addition to the design of testing and evaluation of the system.

In chapter 5, the implementation of the system is explained, going into more detail about how the chosen design was created and put into action. Chapter 6 discusses the results of the system along with the evaluation of the performance of the system.

Finally, chapter 7 details the future improvements that could be made to the system to improve accessibility and usability of text mining in the medical field.

Chapter 2

Literature Survey

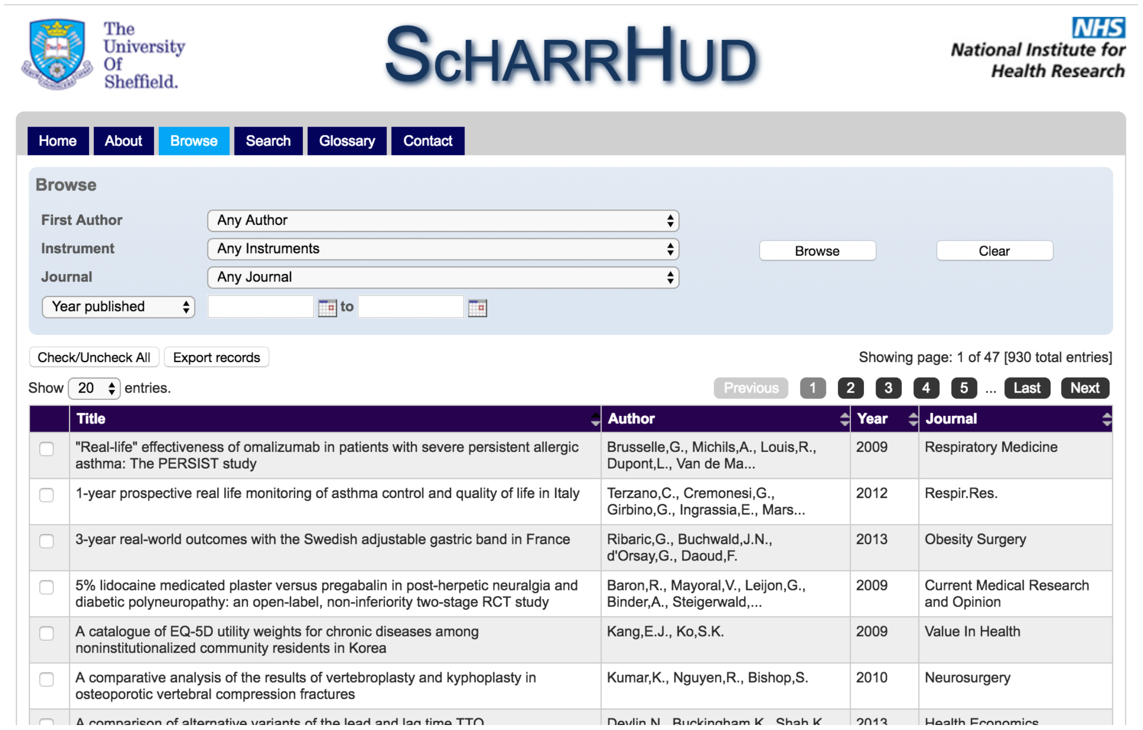
2.1 Medical study screening

The medical study screening process is most commonly undertaken as part of a systematic review. This is a process which aims to answer a defined research question by collecting and summarising all empirical evidence that fits a pre-specified eligibility criteria [3]. Within the field of medical research, this generally involves building a large boolean search query and running it against the PubMed database to retrieve a list of relevant medical studies [4]. The issue, however, is that the resultant list of studies can be extremely large and is not ranked in any particular order. It then becomes the task of experts to manually read each of the documents in the list and confirm its relevance to the systematic review [5]. This can expend a substantial amount of resources and is time consuming as many irrelevant studies need to be examined in order to identify those which are the most useful [6].

2.2 ScHARRHUD: Aims and objectives

ScHARRHUD is a database created by the School of Health and Related Research at the University of Sheffield [7]. The database focuses on collecting the details of studies that are reporting health state utility values (HSUVs) in order to improve access and availability to research and evidence surrounding the subject area [8].Health state utility values are single values, commonly on a scale from 0 to 1, which represent the general health or health-related quality of life of a patient. As well as storing the bibliographic details of the studies, it also extracts and indexes the names of all instruments used in the studies. ScHARRHUD has become a publicly available resource and one of its aims is to rapidly expand its content set. At this stage, all identification of relevant studies has been achieved through a manual process resulting in major strain on available resources [9]. The next step is for the School to investigate ways to quickly improve the quality of the data collection whilst also making more effective use of the available resource. [10]

At present, the database contains 930 entries from the years 2008 to 2013. Each entry is made up of its reference type, title, source, author, abstract, year of publication, keywords, language, PubMed id, web URL, record number, volume, issue, relevant pages and instruments mentioned within the document. The current set of documents in the database is based on a previous search on the EQ-5D instrument. The EQ-5D instrument is a preference based health related quality of life measure which evaluates the generic quality of life of a patient [11].



First Author: Any Author
Instrument: Any Instruments
Journal: Any Journal
Year published: to

Showing page: 1 of 47 [930 total entries]

Show 20 entries.

	Title	Author	Year	Journal
☐	"Real-life" effectiveness of omalizumab in patients with severe persistent allergic asthma: The PERSIST study	Brusselle,G., Michils,A., Louis,R., Dupont,L., Van de Ma...	2009	Respiratory Medicine
☐	1-year prospective real life monitoring of asthma control and quality of life in Italy	Terzano,C., Cremonesi,G., Girbino,G., Ingrassia,E., Mars...	2012	Respir.Res.
☐	3-year real-world outcomes with the Swedish adjustable gastric band in France	Ribaric,G., Buchwald,J.N., d'Orsay,G., Daoud,F.	2013	Obesity Surgery
☐	5% lidocaine medicated plaster versus pregabalin in post-herpetic neuralgia and diabetic polyneuropathy: an open-label, non-inferiority two-stage RCT study	Baron,R., Mayoral,V., Leijon,G., Binder,A., Steigerwald,...	2009	Current Medical Research and Opinion
☐	A catalogue of EQ-5D utility weights for chronic diseases among noninstitutionalized community residents in Korea	Kang,E.J., Ko,S.K.	2009	Value In Health
☐	A comparative analysis of the results of vertebroplasty and kyphoplasty in osteoporotic vertebral compression fractures	Kumar,K., Nguyen,R., Bishop,S.	2010	Neurosurgery
☐	A comparison of alternative variants of the lead and lag time TTO	Devlin,N., Buckingham,K., Shah,K.	2013	Health Economics

Figure 2.1: The ScHARRHUD database

2.3 Text mining

Text Mining is the process of retrieving information from text by transforming it into data. Through the application of natural language processing, text mining can be used in a variety of areas such as information retrieval, pattern recognition and document similarity [12].

There are two main factors which make text mining a particularly successful technology. Firstly, the speed at which text mining is able to produce results is far beyond that achievable by manual means. A text mining application maximises the speed of computing to perform simple arithmetic and produces accurate results in a fraction of the time in comparison to human efforts. In addition, text mining can be applied to huge datasets, such as the MEDLINE database, a task beyond the capacity of any human.

Text Mining has already played a significant role within the medical field due to the large amounts of textual information which exist in this area. The search engine GoPubMed has used text mining to great advantage for the Gene Ontology (GO) and Medical Subject Headings (MeSH) to apply a structure to the large collection of papers in the MEDLINE database [13].

2.4 Text mining techniques

2.4.1 Table of definitions

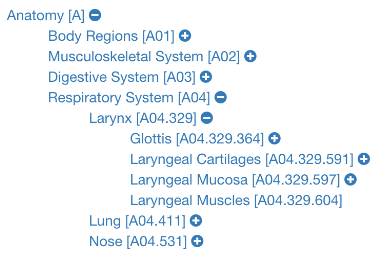
The table below details some of the terms used throughout the following sections in order to explain the technical background of text mining.

Term	Definition
Index	Dictionary of terms detailing which document they appear in and frequency of appearance in that document
D	The collection of documents
$\|D\|$	The size of the collection of documents
$tf_{w,d}$	Term frequency, the frequency of w in document d
cf_w	Collection frequency, the frequency of w in the collection
df_w	Document frequency, the number of documents that contain the term w

Figure 2.2: Text processing definitions

2.4.2 Indexing

Indexing is the process of finding terms that describe documents. There are currently both manual and automatic approaches to the implementation indexing although each process is quite distinct. Manual indexing is useful for closed collections and achieves results with high precision. However, in this situation, the labeller needs to be well trained, otherwise potential problems can result from inconsistent labelling. One of the largest sets of manual indexing is medical subject headings (MeSH) which contain a large set of terms to index medical documents. This index proves a hierarchical structure, becoming more detailed as the structure develops but results in a significantly more precise index.



- Anatomy [A]
  - Body Regions [A01]
  - Musculoskeletal System [A02]
  - Digestive System [A03]
  - Respiratory System [A04]
    - Larynx [A04.329]
      - Glottis [A04.329.364]
      - Laryngeal Cartilages [A04.329.591]
      - Laryngeal Mucosa [A04.329.597]
      - Laryngeal Muscles [A04.329.604]
    - Lung [A04.411]
    - Nose [A04.531]

Figure 2.3: Example of medical subject headings (MeSH) [1]

While manual indexing takes advantage of a predefined vocabulary for its index terms, automatic indexing does not require human input. Instead, it uses the natural language within the documents themselves as its index. Given a collection of documents, for each term, it will record the document in which it appears and the frequency of the occurrence.

another	13:2 14:1 5:1
begin	8:3 7:2
conduct	4:4 1:2 7:1

Figure 2.4: Example of a section of an index

These indexes are extremely useful for then defining documents within the collection as vectors against which computations can be run.

2.4.3 Vectorization

As mentioned previously, converting text into data is the main method used in text processing. This section will review the techniques that make this possible. Documents are converted from text into a high-dimensional vector. Each term within the index relates to a dimension of the document vector and the value of the dimension relates to the frequency of that term in the document or alternative term weighting value. In addition to the documents being converted into vectors, the query/search is also converted into a vector so that a similarity calculation can be run between each document and the query.

	doc_1	doc_2	doc_3	doc_4	doc_5
$term_1$	0	1	0	1	1
$term_2$	0	1	0	0	1
$term_3$	1	1	1	1	0
$term_4$	1	0	1	0	1

Figure 2.5: Example of binary document vectors

A standard term weighting scheme is the tf.idf term weighting method. Tf.idf involves taking the term frequency within a document and multiplying it by the inverse document frequency. This term weighting scheme is based on the concept that the most useful terms for finding relevant documents are the ones which appear less frequently. [14]

$$\begin{gathered} tf_{w,d} = freq_{w,d} \\ idf_{w,D} = \log \frac{|D|}{df_w} \\ tf.idf_{w,d,D} = tf_{w,d} * idf_{w,D} \end{gathered}$$

Figure 2.6: tf.idf equations

2.4.4 Preprocessing

There are two main methods used in the preprocessing stage of text processing; stemming and stop word removal. These methods are applied to the documents in order to improve the final results.

Stemming is the process of conflating morphological variants of words by removing their affix. For example, the words ‘measuring’, ‘measured’ and ‘measurement’ will all become ‘measur’ after applying stemming. This is an effective technique to both reduce computation and improve results as it generalizes a collection of words into their true meaning.

The other main method used in preprocessing is stop word removal. One issue associated with basic text mining, is that it fails to identify the most informative words and treats words such as ‘the’ to be of equal value to other more informative words such as ‘heart’. A stoplist

measuring
measured
measurment

Figure 2.7: Example of stemming

contains a list of words that provide little or no information in assessing the document. These are words such as ‘the’, ‘and’, ‘is’ etc. Performing stop word removal is the process of removing all of the words in the stoplist from the documents in order to leave only those that are the most informative. At this stage, the vectors created from the processed documents result in a much better description of the true meaning of the document and help to improve results when searching for similar documents.

2.4.5 Cosine similarity

Cosine similarity is a measurement in vector space that describes the relative similarity between two vectors [15]. The metric looks at the cosine of the angle between two vectors and then divides this by their lengths to produce a number between -1 and 1 based on how similar the vectors are. As the query and documents can all be represented as vectors, this metric can be used to assign a numerical value to the similarity between query q and document d with the formula below. The cosine similarity equation normalizes the query and document vectors and then finds the angle between the two where a small angle represents a high similarity and a large angle representing a low similarity.

$$sim(q,d) = cos(q,d) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}}$$

Figure 2.8: Cosine similarity

2.5 Text mining evaluation

There are three main metrics for evaluating the performance of an information retrieval system such as a document ranking system; precision, recall and F-measure. All of these measures relate to the relevance of the documents returned by the system. One disadvantage of using these evaluation techniques is that a pre-created gold-standard list must be used which comparisons can be made against. Recall relates to the proportion of relevant documents returned by the system. Precision relates to the proportion of retrieved documents that are relevant. F-measure is a method to combine both precision and recall into a single figure. Precision and recall have equal weights in the f-measure but the score is penalised if one has a particularly poor performance. Each of these metrics produce values between 0 and 1, where 0 represents a system unable to return any relevant documents and 1 represents a system that returns the perfect set of relevant documents.

	Relevant	Non-relevant	Total
Retrieved	A	B	$A+B$
Not Retrieved	C	D	$C+D$
Total	$A+C$	$B+D$	$A+B+C+D$

Figure 2.9: Relevance against retrieved document table

$$Recall = \frac{A}{A+C}$$

$$Precision = \frac{A}{A+B}$$

$$FMeasure = \frac{2*Precision*Recall}{Precision+Recall}$$

Figure 2.10: Precision and recall equations

2.6 The use of text mining in medical paper identification

Over the past 30 years, the medical field has started to investigate the use of text mining much more robustly [16]. Whilst it is still a developing area of research, it has already been found to provide benefits for medical study identification [17]. There are currently three main avenues in which text mining is used within the systematic review/medical document review field:

- Automatic classification, where a document is categorized based on its contents. For example MeSH terms can be applied to a document [18].
- Document summarization, where the most informative parts of a document are extracted and used to form a summary of its contents.

- Document clustering, where similar documents are identified and grouped into clusters [19].

When attempting to identify relevant documents as part of a systematic review, reviewers frequently use searches of a range of online databases such as MEDLINE, PubMed, Google Scholar etc. In order to retrieve a good range of results without publication bias, reviewers often have to sacrifice the precision of the search leading to a much larger number of results being return by the search. While text mining can be used after the search to reduce the burden of sifting manually through the results, it could also be used to identify the best search terms in advance of performing the initial search. If previously relevant documents have been identified, then text mining could then be used to extract the most informative terms from these documents that can be used as search terms in order to find similar studies. By applying these text mining techniques before the search, the chances of finding relevant documents to the search will be maximised.

Once a search has been completed, reviewers are faced then with the screening process, when the returned documents must be manually sifted. A significant issue here is that the search returns documents in no particular order, meaning that reviewers would have further work to ensure that relevant documents are not omitted. One method that can be employed at this stage is ‘screening prioritization’, where particular documents are paced at a higher priority based on particular variables. Text mining is more sophisticated in that, during the prioritisation process, it can make use of term recognition and similarity values to the search query to produce a ranked list of documents. This saves time and effort as the more relevant documents can be screened first and the most relevant information is identified sooner in the process.

2.7 Accessibility of text mining to systematic reviewers

A current issue within the medical field is that running text processing systems may seem overly technical to systematic reviewers. Progress could be made by creating a reusable and user-friendly tool. This would include a graphical user interface, changing from the current method of running command line scripts and the ability to change the collection of documents to be used as the query. Production of this tool would make the use of text mining more accessible to systematic reviewers and be a significant step forward in this area.

Having researched the existence of such a tool, two publicly available text mining tools have been identified; PubTator [20], a web-based tool for medical literature curation and Anni [21], a document retrieval system within the biomedical field. Both however still seem to require background knowledge on the part of the user in order to operate. The creation of a user-friendly system would increase the scope of text mining to make it more accessible to a larger audience.

2.8 Comparison of manual screening and text mining techniques

While there has been considerable research into the advantages and disadvantages of the application of text mining, the degree to which text mining adds value to the quality of the results is yet unquantifiable. It is evident that text mining techniques are advantageous when implemented correctly as part of a systematic review, but it is also clear that further research is needed in the area. There is speculation around its accuracy and, more specifically, the possibility that human interaction may be more accurate in avoiding omissions or misclassification of documents.

The O'Mara-Eves et al. (2015) review [22], found that most studies implied that a workload reduction of between 30 to 70 percent is possible but at present, there are insufficient comparison metrics and evaluation of the difference is difficult. The review did however suggest three conclusions relating to the use of text mining in systematic reviews;

1. The use of text mining to prioritise the order in which items are screened should be considered as safe for use in live reviews.
2. The use of text mining as a second screener may also be used with caution.
3. The use of text mining to eliminate studies automatically should be considered promising.

The statement that 'the use of text mining in prioritising the order of documents to be screened is considered to be safe' is a significant step forward for the field as it provides confirmation that text mining is advantageous. Previous to this, most papers had stated that text mining could be useful but none had been able to confirm that it was of definite benefit.

2.9 Active learning

Active learning is a machine learning technique in which the system iteratively queries a user to obtain outputs and new data points [23]. The system generates a small set of new data points based on its current training set and asks the user to supply the output values for them. Once these new points have been labelled, they are added to the training set and the system then generates additional new points. As a result the training set improves with each iteration and is therefore able to generate improved results over time [24].

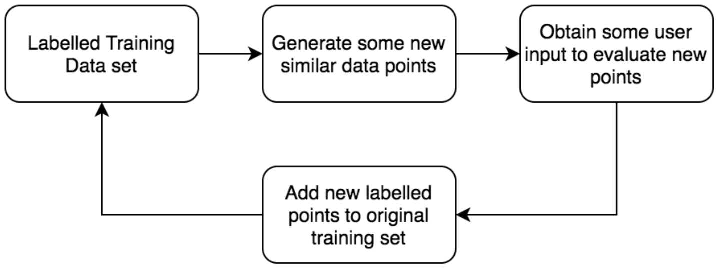

Labelled Training Data set
Generate some new similar data points
Obtain some user input to evaluate new points
Add new labelled points to original training set

Figure 2.11: Active learning flow chart

2.10 Relevance feedback

Relevance feedback is an application of active learning in the text mining field. The idea is that the query is updated and improved with each iteration in order to improve the effectiveness of the system [25]. Relevance feedback uses the results from an initial query being run through the system combined with some 'relevance judgements' from a user or automatically to create an improved query which can be run through the system again. This standard variation of relevance feedback is frequently implemented with the Rocchio algorithm. The Rocchio algorithm [26] makes use of user relevance judgements to split documents into two sets; relevant and non-relevant. It then uses these two sets to add terms from relevant documents to the query and remove or reduce the weights of terms included in non-relevant documents.

This implementation of relevance feedback however does require user input in making the relevance judgements and to avoid this, this project will implement pseudo relevance feedback. Pseudo relevance feedback, also known as blind relevance feedback, is an implementation in

$$q' = \alpha q + \beta \frac{\sum R}{N_R} - \gamma \frac{\sum C}{N_C}$$

q' : New query vector
q : Original query vector
R : Set of relevant document vectors
C : Set of non-relevant document vectors
N_R : Number of relevant documents
N_C : Number of non relevant documents
α, β, γ : Rocchio weights (constants)

Figure 2.12: Rocchio algorithm

which no user input is required. The algorithm works by assuming that the top N ranked documents are relevant and adding the top T terms from the set N to the original query.

2.11 CLEF 2017

CLEF (Conference and Labs of the Evaluation Forum) eHealth is an annual lab aimed at evaluating the current eHealth information extraction and information retrieval technologies. CLEF brings together researchers, providing them with datasets and releases a set of tasks which participants can attempt to solve. In 2017, the University of Sheffield took on the second task offered by CLEF on the topic of technology assisted systematic reviews in empirical medicine, which focused on the ranking of studies for a review [27]. The representatives of the university built and submitted a information retrieval system in response to the task. This system used various combinations of the document title, abstract and MeSH headings to rank the documents. The code for this system is examined later in the report - Section 3.2 contains some discussion of this issue to establish if it can be adapted for use in this project.

2.12 Summary

The use of text processing with the medical field is of growing interest and is beginning to catch the attention of experts in the area. Whilst previously there was scepticism about the accuracy of results, more evidence is beginning to surface regarding the advantages of text mining in both reducing bias and workload. Given that this area of research is still active and developing, it is argued that progress would ensue if text mining is made more accessible to reviewers.

Chapter 3

Requirements and analysis

This section will cover the requirements of the project. The main requirement of the project was to find other relevant documents that can be added into the ScHARRHUD database. This is achieved through creating a system to complete the preprocessing, vectorization and similarity calculations that are required to produce the ranked list of documents.

3.1 Project requirements from ScHARR

ScHARR are looking to expand the ScHARRHUD database but are struggling on time and resources to do so. They are looking to add more studies to that database that contain information regarding health state utility values. ScHARR's focus in on the improving the collection of relevant documents rather than the procurement of a reusable tool. ScHARR confirmed that the focus of the current database is on documents that include measurements of patient quality of life, with a bias towards EQ-5D instruments. They would like to extend the database to cover other instruments but for now want to focus on adding additional documents which mention the the EQ-5D instrument.

3.2 Clef 2017 code

I was given permission to adapt some previously developed python code which provides an informal retrieval system that ranks documents based on a set of input queries [28]. It was previously developed as part of a text processing competition and makes use of some python libraries to do tf.idf vectorization and cosine similarity calculations [27]. While sections of this code provide an idea of how text mining can be implemented, very little of the code was adaptable for use in this project. In particular, the code wasn't well structured and was difficult to understand. Rather than having a main file to run with flags as options for controlling the script, each different option is set up in an individual file. For example, running the information retrieval system with stemming enabled is a completely different file to running the system without it. Furthermore, this system takes a set of queries as its input whereas the source for this project is a collection of documents from ScHARRHUD which must be converted to vectors.

3.3 Using the current ScHARRHUD data: query vector creation, test and train data

3.3.1 Query vector creation

The first requirement of this project is to find other documents that are similar to those in the existing database. This requires creating a single query vector by processing the existing collection with stoplist removal and stemming to remove uninformative terms and conflate morphological terms. The pre-processing techniques are expected to reduce system run times and improve results while the single vector will be used as the query vector for the new system.

3.3.2 Document vector creation

Before the test documents can be converted into vectors, the index of the test collection must be created. This involves iterating over all of the documents in the collection and for each term adding the document id and term frequency in that document into the index. This produces an index which contains a list of all the terms in all of the documents and for each term a list of document ids and the frequency that the term appears in that document.

The collection of documents is then converted into individual document vectors. Again, these will be preprocessed using stemming and stoplist removal and then vectorized using the tf.idf term weighting scheme.The tf.idf weighting scheme is being used as it penalises terms that are less informative and therefore is able to give a better depiction of what the document represents when converted to a vector.

3.3.3 Performing cosine similarity calculations

The most important part of the project occurs when calculating the cosine similarity between the query vector and the document vectors in the search space. This is where the system will determine the extent to which a document is relevant for inclusion into the ScHARRHUD database. A python library is used in the code to perform these calculations and then the document set was ranked. This produces the final results of the system which can then be passed onto ScHARR for review.

3.4 Project evaluation

The results of this project revolve around interfacing with ScHARR. Text processing evaluation methods such as precision, recall and F-measure require a gold standard dataset to against which to compare system results. However as this is a real world search there is no gold standard dataset. Therefore after running the system and receiving a ranked list of documents, the results need to be manually screened by ScHARR, by reviewing the best documents and assessing their relevance. If ScHARR find that the ranked list has produced relevant documents that can be added into the ScHARRHUD database, then the project can be deemed a success.

Another method of evaluating the performance of the system, is to split the current collection of documents in the database into a training and test set. This is achieved by time slicing the dataset, taking documents from the years 2008-2012 as the training set and using the documents from 2013 as a test set. After running the system, comparisons can be made between the results and the known relevant documents that were added into the test set. Values for the precision and recall of the system can be calculated which give an indication of the performance of the system.

3.5 Active learning and relevance feedback

The implementation of active learning was not an additional object of the project and was dependant on the success of the results from the main system. As active learning requires some human input to evaluate the newly generated data points which were in this case documents, the implementation of this section required some input from ScHARR [29]. In particular, the newly generated documents needed to be manually screened by an expert at ScHARR and given a label based on its relevance to the ScHARRHUD database. Additionally, as active learning is an iterative process, the manual screening step was potentially required several times. Pseudo relevance feedback can however be implemented as it requires no user input and relevance judgements are passed automatically [30].

3.6 Summary

ScHARR are currently working on a project to collect medical literature that report health state utility values and make these publicly available in a database named ScHARRHUD. The requirement of this project was to aid in the discovery of new documents for inclusion in the database. This has been achieved by building an information retrieval system which implements text processing techniques to find documents most similar to those already included in the database. There was also scope to further improve the system by implementing relevance feedback and active learning.

Chapter 4

Design

4.1 Introduction

This section will discuss how the system was designed and why choices of particular structure, languages and features were made. There will be a general overview of how the system is intended to work and why, while the implementation section will detail the code in more depth, explaining how the features were created.

4.2 Design

The system was designed as a collection of python scripts to retrieve the required data, perform operations on it and evaluate the results. The purpose of the system in the project is to provide results and therefore the scripts were designed to be run from the command line with no user interface, requiring the user to have some prior knowledge of command line flags and options.

The CLEF code, developed for a similar type of purpose, created a library of python scripts but duplicated code between files and did not allow the user to easily customise different types of run. For example, there was one script for running the system using stoplist and stemming and another very similar script for running without either of the preprocessing techniques. This project was designed to make user customisation simpler by having a single file to run with command line flags used to control what methods to be executed for a particular run.

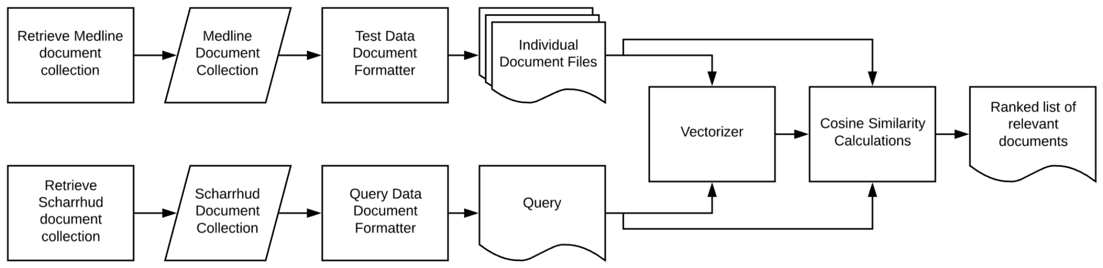

Retrieve Medline document collection
Medline Document Collection
Test Data Document Formatter
Individual Document Files
Vectorizer
Cosine Similarity Calculations
Ranked list of relevant documents
Retrieve Scharrhud document collection
Scharrhud Document Collection
Query Data Document Formatter
Query

Figure 4.1: UML diagram for the design of the system

4.3 Why python?

For such a data intensive project, python was the best choice of language to write the system in. Python has very good support for file handling which would be used throughout the system as well as giving access to a range of libraries that help with data handling, vectorization and Medline data retrieval. Functions for vectorization and cosine similarity were available from the sklearn library and Entrez from Bio library was used to retrieve Medline data for the documents already in the ScHARRHUD database.

4.4 Data structure design

Given that the system would be handling and producing vast amounts of data, it was important to design how the data should be structured. There are two sets of input data, a query and the document collection and one set of output data, the ranked list of documents. The query data was the easiest to design as it consists of a single file into which all of the abstracts for the documents in the SHARRHUD database can be copied. The medline data was only downloadable in large batches in xml format and included a large amount of information relating to each document that was not useful for the text mining task such as year of publication and authors. One option was to read the abstracts and ids directly from the downloaded xml files. However this meant that the document collection lacked accessibility outside of running the system, i.e. if the user wants to read the abstract of a particular document there is no easy method to access the file without writing some code. The document collection was instead formatted into a set of individual text files, one per document, named using the Pubmed Id and the contents being the abstract. This simple structure was adequate for the system to produce meaningful results. Another option for structuring the document collection was to use a database which stored the ID and abstract for each file. This however required development of a formatter to process the documents into the desired data structure. A further option which was only thought of once development had already begun, was the use of a database to store the ID and abstract of all of the documents. This would have been a superior structure to use as the user can easily query the collection as well as add and remove documents. This may also have resulted in quicker run times of the system as querying the database is faster than opening and reading individual files.

For the results, the structure of a ranked list of pairs was used, where each pair was the Pubmed ID of the document and the cosine similarity value for the document.

4.5 Testing design

The main aim of the project was to produce a list of documents that can be presented to ScHARR to add into their database. A natural method for evaluating the results is a count of the number of presented documents that are accepted by ScHARR as being relevant. This however requires some manual checking of each document by an expert at ScHARR and given their stretched resources, it was unlikely that they would be able to offer the time immediately. Therefore an alternative method of testing the system was to split the known relevant data and insert a section of it into the test data and to see if the documents are returned as relevant. As shown in the UML diagram (Figure 6.2), the system splits the ScHARRHUD documents by year of publication, with documents after 2012 being added to the test data and all others being used as the query. Once the ranked list of documents has been produced by the system, evaluation metrics for recall, precision and f-measure can be calculated using the collection of documents added to the test data. These scores will give an indication of how well the system performs at retrieving relevant documents.

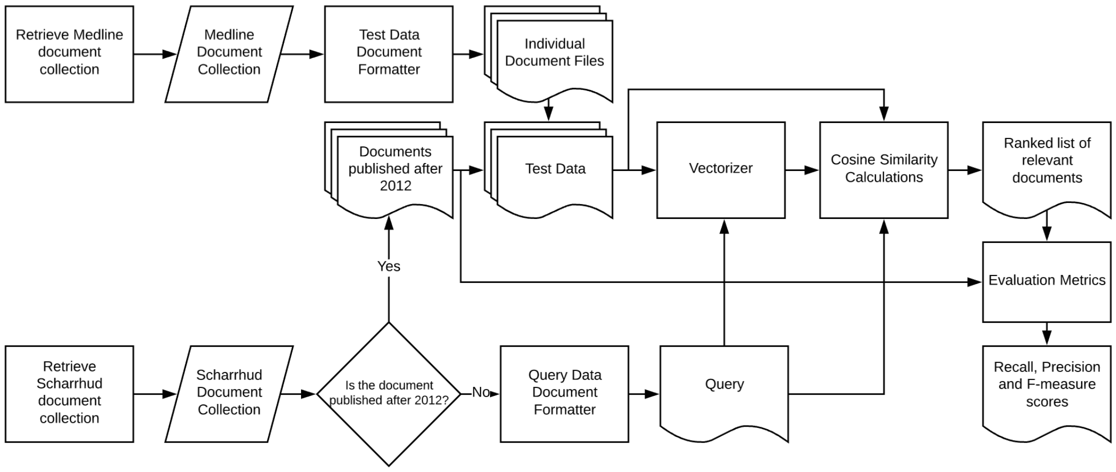

Retrieve Medline document collection
Medline Document Collection
Test Data Document Formatter
Individual Document Files
Documents published after 2012
Test Data
Vectorizer
Cosine Similarity Calculations
Ranked list of relevant documents
Yes
Evaluation Metrics
Retrieve Scharrhud document collection
Scharrhud Document Collection
Is the document published after 2012?
No
Query Data Document Formatter
Query
Recall, Precision and F-measure scores

Figure 4.2: UML diagram for evaluating the system

4.6 Relevance feedback

The implementation of classic Rocchio relevance feedback was made difficult due to user relevance judgements being difficult to obtain. Instead an implementation of pseudo relevance feedback was designed. This involved the top X ranked documents from the initial results of the system being assumed as relevant. This set of documents, R, then had their terms weighted using tf.idf and the top Y terms with the best scores, added to the query. Once the new query had been created, the system iterated over for a user defined number of iterations.

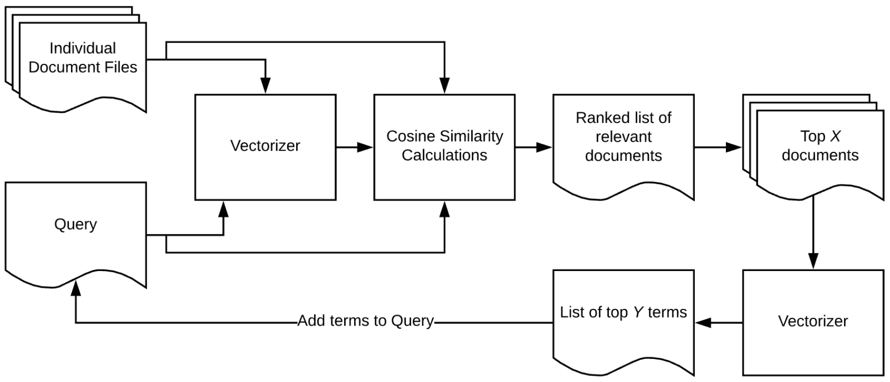

Individual Document Files
Vectorizer
Cosine Similarity Calculations
Ranked list of relevant documents
Top X documents
Query
Add terms to Query
List of top Y terms
Vectorizer

Figure 4.3: UML diagram for pseudo relevance feedback

4.7 Where to run the system

The question of how and where to run the system arose after looking at the volume of data to be handled. Initially, the plan was to use the whole of the Medline Base repository as the test data for the system. However, having accessed the volume of data, it became apparent that the system run times would be too long. The whole repository consists of 892 xml files approximately 25MB each and when formatted contains around 20,000 individual documents. This includes documents published from 1975 to 2017. If all of this historic data was used, the system would need to process approximately 18 million files and would need considerable processing power to complete within a reasonable time. This raised three questions; what hardware would the system run on and whether the data should be reduced and how.

The plan was for the system to be run from a personal laptop but after further investigation, there was the option for it to be run on Iceberg, a high performance server hosted by the University of Sheffield. However, instead of changing the infrastructure of where the system would run, a decision was made to reduce the amount of data being fed into it. Rather than use the whole repository, the test data was limited to only those documents published between 2005 and 2013. Due to the current ScHARRHUD data focusing around the EQ-5D instrument, which was only established in 1990 [11] it was unlikely that earlier documents would be relevant. By setting limits between 2005 and 2013, it reduced the input to around 6 million files and gave an estimated runtime of 6 hours. This meant that the system should be able to run without issue on a personal laptop and would still produce meaningful results.

4.8 Summary

The design of the system focused mainly on data handling and how to best store and process the large amounts of data needed to produce results. The main functions of the system followed a classic information retrieval process; pre-processing the data, vectorizing the documents and performing similarity calculations. Testing for the performance of the system was made difficult as only live data was available with no gold standard dataset. However, this was overcome by time-slicing the known relevant document set and adding a subset to the test data.

Chapter 5

Implementation and Testing

This section will outline how the chosen design was implemented and detail in depth about how the code was built and why. The system was written in Python and designed to be run from the command line with a focus on user customisation.

5.1 Implementation

5.1.1 Setting up

The system is hosted in a Github repository for easy access and cloning of fresh copies. Once downloaded, there is some initial set up required which is done by using the build script. This script creates the recommended folder structure used for housing the data.

5.1.2 Acquiring the data

For the system to function, it must have two sets of input data: a query and the test data. In order to obtain the query, a download was made of the ScHARRHUD database in 'csv' format from their website. From here, a list of the document Pubmed ID's was made and stored in a simple text file called `PM_ids.txt`. The system uses the list and a function called `fetch_query_abstracts` which retrieves the abstracts for each document and enters them all into a file called `query.txt` which will act as the query for the system.

Acquiring the test data and deciding how to store it was difficult due to the amount of data being used. The script `download_medline_repo.py` was written to handle fetching the medline files from the baseline repository. The script gives an option to define a path for the location of the downloads, without which it will default to the `Scharrhud_Data/medline/` folder which was created during the build process. Having retrieved the medline files, they are then formatted from their xml format into the individual text files that are required for the system. The `format_medline_files.py` script was built to handle the conversion process. The retrieval of the data and its formatting were split into two individual scripts rather than doing both at once as each process has long run times. This enables users to run the scripts individually in smaller time chunks.

These scripts were both written to enable users to decide where they wanted the data stored rather than hard coding paths to specific folders. As a result, the system could be run on a server and data could be stored on an external hard drive.

5.1.3 Core system script

The core script which implements the text mining techniques is the `sharrhud_identification.py` script. This script takes the prepared test data and query and applies pre-processing before vectorizing the whole collection using the `TfIdfVectorizer` function from the sklearn module. Having computed the tf.idf matrix, the system applies the `linear_kernel` function also from sklearn to find the cosine similarity values for each document. Sorting this list of values gives the desired result of a ranked list of documents in order of similarity to the given query.

5.1.4 Command line flags and customisation

This system aims to include a level of customisation and reusability for the scripts. In order to accommodate different types of runs, a range of options are presented to the user when running the script. This was achieved by using the getopt library and creating a command line class in the code from which a config could be created and passed to the rest of the system. The table below details the possible flags implemented in the system and a description for each.

Flag	Definition
-h	Prints help for the system to the console. Details the flag options.
-q	Control whether to retrieve a new query file. This generally will only need to run on the initial set up as once a query file has been created, it will not need to be made again.
-o	Define a path to an output file where results can be stored. Allows for results from different runs to be stored in multiple files.
-r	Define the number of results to be stored. The number of the most relevant files to be included.
-p	Define a path to the desired test data.
-m	If the system has been run before, storing the vectorizer and tfidf matrix then this gives the option to preload these objects. This can dramatically reduce runtime.
-s	Define a cut off for the similarity values for the results. For example only return results with a similarity value greater than 0.2.

Figure 5.1: Command line options

5.1.5 Run time reduction

While building and running the system, it became obvious that performing computations on large datasets resulted in long run times for the scripts. With only one hundred thousand documents in the test data, it took six and a half minutes to run the system. Each time the script ran, all documents would have to be loaded and then the collection vectorized before the similarity calculations could be performed. Given this, a novel solution was found for being able to save and store the vectorized representation of the document collection, meaning that this time-consuming part of the script had to be completed only once instead of on each run. Once it had initially vectorized the document collection, the vectorizer was stored in a pickle file while the tfidf matrix was stored in a npz matrix file. Both of these could be loaded back into the system for reuse at any time.

An experiment to test the effectiveness of the storage technique on the runtime of the system was performed. Two collections of documents would be used for input with one containing 100512 individual files and the other containing 15848. The run times were calculated using the `timeit` package included in python. The amount of storage required was also examined to see if the proposed method would reduce the necessary memory. The 'Calculated' run is when the system doesn't pre-load the matrix or vectorizer and the storage is the total sum of the size of the individual files. The 'Pre-loaded' run is when the files are loaded into the system from files and the storage is the size of the vectorizer pickle file and the matrix file combined.

	15878 Documents		100512 Documents	
	Run Time (min)	Storage (MB)	Run Time (min)	Storage (MB)
Calculated	1.30062	15.2	7.37067	95.2
Pre-loaded	0.13285	15.7	0.14761	91.5

Figure 5.2: Run time and storage required experiment results

For the smaller collection of documents, run time was reduced from 1 minute and 18 seconds to approximately 8 seconds amounting in a total reduction of 1 minute and 10 seconds. However, the storage required increased from 15.2 MB to 15.7 MB as the matrix and pickle file used more memory. However, as only a small amount of storage was initially required, the increase is negligible. For the larger collection of 100512 documents, run time decreased significantly from 7 minutes and 22 seconds to only 8.4 seconds. This results in an approximately 5200% reduction in run time and a total fall of 7 minutes and 29 seconds. Additionally, as the size of the document collection grew, the storage required for the pickle and matrix files fell below the amount required for the individual files. Further experiments should be done in this area to investigate at the effects on much larger document collections. One observation is that pickle may not be able to deal with storing and reading with much larger files.

One possible recommendation is for the Medline Base Repo [31] to distribute a vectorized format of each baseline repository alongside the zip files they already host. This would increase both awareness and access to text mining for the medical field.

5.1.6 Output

The system produces results in the form of a ranked list of documents in order of their similarity to the query. Using the '-o' flag, results can be stored into a text file with a name defined by the user enabling them to be saved and used in the future. Additionally, the '-r' flag can be used to define the number of results the user wishes to save. The structure of the results is as show in the figure 5.3, where each line contains the rank of the document, the name of the document and the similarity value it scored. Another customisation available for the results is to use the '`-s`' flag which can set a cut-off for the similarity values of the results. For example, the user may only want to return results that have a similarity value greater than 0.3. This flag is an exclusive or to the'`-r`' flag.

```
1: 17893863.txt 0.4958306082461012
2: 17999424.txt 0.49412548455926936
3: 17922304.txt 0.42783662524398397
4: 17888106.txt 0.38304705845740084
5: 17888587.txt 0.3776498618684394
```

Figure 5.3: Example of results obtained by the system

5.1.7 Relevance feedback implementation

As a further experiment to improve the system, an adaptation of the main core script was created which included an implementation of the previously discussed pseudo relevance feedback (See section 2.10). As well as having all of the command line flags implemented in the core script, an additional '`-i`' flag was added which gave the user an option to define the number of iterations of relevance feedback to use.

In this implementation of the text mining system, a loop is placed around the vectorization and comparison of the query and test data and an additional `improve_query` function is called after the results are ranked. The `improve_query` function takes the top N ranked documents and performs tf.idf calculations on this set. Having done so, it takes the top T terms based on their tf.idf and adds them to the original query before passing the new query back into the system for the next iteration.

An interesting metric to look at when using relevance feedback is the difference in two sets of results based on the number of iterations or the number of terms T added to the query. To obtain this metric, a `results_difference.py` script was created which makes use of the `SequenceMatcher` function from `difflib` to return the percentage difference between two sets of results.

5.2 Testing the system and evaluating results

Alongside the core `scharrhud_identification.py` script, two other types of system run were created. Firstly a script to evaluate the performance of the system's precision and recall was produced and called `split_query_data.py`. In this run, the documents known to be relevant are split based on their year of publication, with documents published after 2012 being added to the test document collection and others being used as the query. By adding a subset of known relevant documents to the test data and running the system on this collection, metrics for the precision and recall of the system can be calculated by looking at how many of the relevant documents are identified. An `evaluate_results.py` script was written which takes two arguments, a gold standard list and a results list, and compares the two to calculate precision, recall and f-measure scores.

Chapter 6

Evaluation

Evaluating the success of this project relies heavily on the feedback from experts in ScHARR. Given their scarce resources and time pressures, obtaining feedback on the results has been difficult. Feedback was given by an expert at ScHARR for the top 20 documents returned by the system from a test run on a small collection. Evaluation of the performance of the system was gauged by a number of experiments where some of the known relevant documents were added to the test data and information retrieval metrics were calculated.

6.1 Evaluation of the effectiveness of the system

To evaluate the effectiveness of the system, an experiment was set up in which some of the known relevant documents were added to the test data. In doing so, figures for the precision and recall could be calculated and these would give an insight into the performance of the system. The F-measure score is used to evaluate the system. This is a metric which produces a value between 0 and 1 which represents the general performance of the system in relation to precision and recall. Usually, testing the performance using precision and recall of a text mining system such as this, is achieved by testing the classifier with a completely labelled data set. The data set should include both positively and negatively labelled data; in the case of this project, both known relevant and irrelevant documents. The issue that arose when testing this system is that there is only a set of positively labelled documents (the documents from ScHARRHUD) and a set of unlabelled data (the documents from Medline) meaning that there are no known irrelevant documents. Therefore when evaluating the precision and recall of the system, all of the unlabelled documents were assumed to be irrelevant even though the set could potentially contain documents with high similarity to the query. As a result, precision and recall values are likely to be lower than values used with traditional models.

For the evaluation experiment, the set of relevant documents was split by year of publication, with 247 documents being added to the test data and the 678 remaining documents being added the query. This represented an approximate 27% / 73% split of the training data while most traditional models recommend a 30% / 70% split [32]. The unlabelled test data consisted of 50353 documents and when combined with the added 247 known relevant documents summed to a total of 50600 documents. This results in a test data set with only 0.4% of documents being labelled as relevant and 99.6% as irrelevant.

Figure 6.1 shows a selection of the results for the performance of the system at variable similarity thresholds. Definitions and explanations for recall, precision and F-measure can be found in section 2.5.

Similarity threshold	Total results	Relevant documents retrieved	Recall	Precision	F-Measure
0.3	48	36	0.1457	0.75	0.244
0.26	88	46	0.1862	0.5227	0.2746
0.25	105	52	0.2105	0.4952	0.2954
0.24	181	73	0.2955	0.4033	0.3411
0.2	408	95	0.3846	0.2328	0.29
0.15	1591	140	0.5668	0.0879	0.1523
0.1	6397	185	0.7489	0.0289	0.0556

Figure 6.1: Run time and storage required experiment results

Given the large amount of irrelevantly labelled documents in comparison to those which were relevant, the system performed very effectively. When testing with only positive and unlabelled data, it is assumed that F-measure scores will be lower than traditional testing models. Additionally, as this is using unlabelled data, the documents returned that are counted as 'false positives' in these calculations appear to be reasonably relevant. 'False positives' are documents that have been labelled by the system as relevant, but are actually not relevant. From an objective viewpoint however, when looking at the abstracts of the 'false positives', they appear to be similar to the ScHARRHUD documents. The three highest ranked "false positives", (PMIDS: 17893863, 17922304, 17879012) all focus on health related quality of life instruments, with interest in their comparison and application. These documents appear to be perfect candidates for addition into the database even though they account for decreases in the evaluation of the performance of the system.

The graph in figure 6.2 shows the plot of F-measure against similarity threshold with the threshold increasing in increments of 0.01 between the values of 0.1 and 0.35. For the best F-Measure score, a similarity threshold of approximately 0.24 should be set. This information could be used for the full system run to obtain the best possible results.

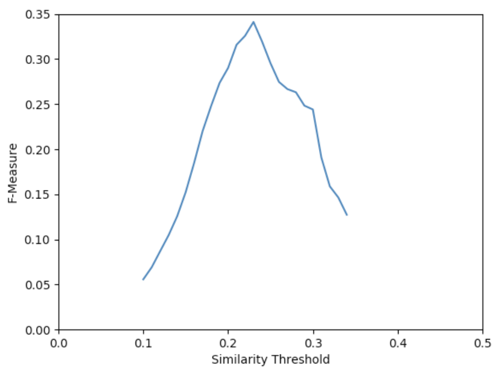


Figure 6.2: F-measure scores plotted against similarity threshold

The table in figure 6.3 shows the performance of the system when taking the top X results as relevant. As the number of relevant results increases, recall steadily grows while precision falls. F-measure improves with each run until the 175 test where it begins to fall again. It falls steadily until 275 where it has a small increase before continuing to decline.

Number of results	Recall	Precision	F-Measure
25	0.08	0.8	0.147
50	0.149	0.74	0.249
75	0.178	0.586	0.273
100	0.206	0.51	0.293
125	0.23	0.456	0.306
150	0.267	0.44	0.332
175	0.295	0.417	0.345
200	0.299	0.37	0.331
225	0.311	0.342	0.326
250	0.319	0.316	0.317
275	0.344	0.309	0.325
300	0.352	0.29	0.318
325	0.352	0.267	0.304

Figure 6.3: Performance of the system with varying number of results returned.

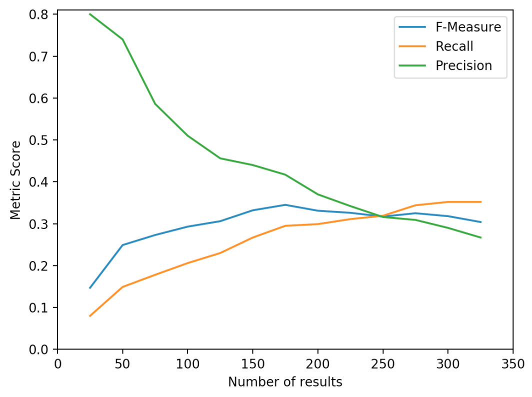


Figure 6.4: Graph showing the performance of the system based on the top number of results assumed relevant

6.2 Effects of stop word lists on the performance of the system

A further experiment was carried out to determine the effect of various stop word lists on the performance of the system. By measuring the F-Measure score of the system while using a variety of stop word lists, the best list can be found and used for the full system run. Four stop word lists were found [33] [34] [35] and used along with a test using no stop word list. For each run a similarity threshold was set at 0.3 and the same data split was used as in Section 6.3.

Stop word list name	Number of stop words	F-Measure
No list	0	0.1672
'english' [35]	319	0.2491
PubMed Stoplist [34]	365	0.244
Large list [**?**]	666	0.2372

Figure 6.5: Performance of the system using different stop word lists

Running the system with the four variations of stop word list, it is clear that using no stop word list greatly reduces the performance of the system. The stop word list with the most superior performance was the 'english' list built into the sci-kit learn TfIdfVectorizer which is taken from the University of Glasgow's Information Retrieval group.

6.3 Evaluation of results by ScHARR

The principal method of evaluating the system is to have the results screened by an expert at ScHARR and to examine the number of results that would be accepted as relevant. A list of the top 20 results from the test run were sent to Suzy Paisley, an expert at ScHARR, to give relevance judgements on each. From the 20 results examined, 15 were deemed relevant and 5 not relevant, giving a 75% precision rate. The five documents were judged not relevant as these did not provide data from preference-based or utility instruments. The 75% precision is a high performing rate for an information retrieval task and suggest that the system is feasible to find studies on a larger scale. Additionally, given that these results came from a restricted set of test data, the fact that the system achieved a 75% rate is very promising for the identification of more documents on a larger scale.

The relevance judgements gained from the expert could then be used to apply relevance feedback to the system using the Rocchio algorithm (See Section 2.10) to further expand the query and improve the results.

6.4 Evaluation of relevance feedback

The performance of pseudo relevance feedback is based on three variables: the number of iterations; the number of terms added to the query; and the number of documents assumed as relevant. Experiments were set up to test how each of these variables affects the performance of the system. All experiments used the same data split as previous tests as well as a similarity threshold of 0.2.

6.4.1 Number of iterations

The first variable tested was the number of iterations of relevance feedback used. The values for the number of documents assumed as relevant and the number of terms added were fixed at 25 and 200 respectively. This experiment tests whether the use of relevance feedback generally improves the results of the system, while changing the other variables tests how to best tweak relevance feedback to get the most effective improvements. The baseline value shows the performance of the system without any relevance feedback implemented.

No. of iterations	No. of documents assumed relevant	No. of terms added	F-Measure
Baseline	0	0	0.29007
20	25	200	0.29709
30	25	200	0.29808
50	25	200	0.30482
100	25	200	0.31544
200	25	200	0.32954
300	25	200	0.34188
400	25	200	0.33333

Figure 6.6: Performance of relevance feedback while changing the number of iterations

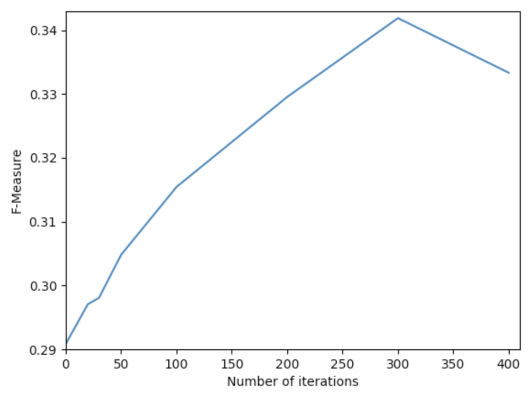


Figure 6.7: Graph showing the F-Measure scores plotted against the number of iterations of relevance feedback

As shown in the table in figure 6.6, the implementation of relevance feedback made a significant improvement to the performance of the system. Additionally, increasing the number of iterations made continued to improve performance until around 300 where the F-measure values began to fall. The number of iterations that achieved the best F-measure was 300, with a score of 0.34188. This was an increase of 0.05181 compared to the baseline performance of the system. From the results, the increase in performance mainly came from removing 'false positives', increasing precision as opposed to improving recall.

6.4.2 Number of terms added to the query

Another experiment was set up to test the effect of the number of terms added to the query with each iteration. Values for the number of iterations and the number of documents assumed as relevant were fixed at 20 and 25 respectively. Figure 6.8 and 6.9 shows the results for each test when changing the number of terms added, in a table and plotted on a graph respectively.

No. of iterations	No. of documents assumed relevant	No. of terms added	F-Measure
Baseline	0	0	0.29007
20	25	10	0.29493
20	25	20	0.29141
20	25	50	0.29675
20	25	75	0.29675
20	25	100	0.29583
20	25	150	0.29447
20	25	200	0.29709
20	25	250	0.29808
20	25	300	0.29357
20	25	350	0.29357

Figure 6.8: Performance of relevance feedback while changing the number of terms added to the query

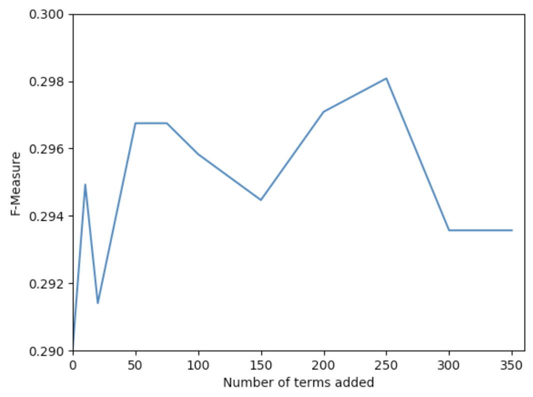


Figure 6.9: Graph showing the F-measure scores plotted again the number of terms added to the query

The performance of the system based on varying the number of terms added gave volatile results. There was no clear correlation, with the F-measure values fluctuating with each test.

6.4.3 Number of documents assumed as relevant

To test the effect of the number of documents assumed as relevant on the performance of relevance feedback implementation, a similar experiment was used. Again, fixing the number of iterations at 20 and the number of terms added at 200.

No. of iterations	No. of documents assumed relevant	No. of terms added	F-Measure
Baseline	0	0	0.29007
20	5	200	0.29102
20	10	200	0.29102
20	15	200	0.29275
20	20	200	0.29538
20	25	200	0.29709
20	30	200	0.29663
20	35	200	0.29402
20	40	200	0.29402
20	50	200	0.29618
20	60	200	0.29268

Figure 6.10: Performance of relevance feedback while changing the number of documents assumed as relevant

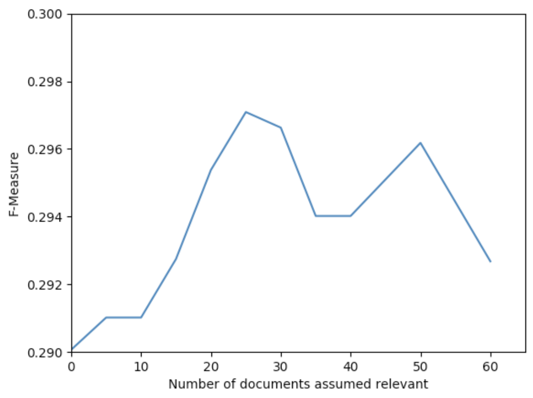


Figure 6.11: Graph showing the F-measure score plotted against the number of documents assumed relevant

As shown in figure 6.11, the system performed best when assuming 25 documents to be relevant. The F-measure grew steadily between the values of 0 and 25 before falling again. At 50 documents assumed relevant, there was a small spike but the downward trend continued afterwards. This is expected as with each document assumed relevant, they become less similar to the query. This also means that they include terms which are less similar and some of these unwanted terms will be added to the query. This causes a shift of the query vector away from the true vector.

6.5 Summary

The results of the project were very positive. The system achieved good base performance scores in testing and the adaptations made to the system such as relevance feedback helped to improve scores further. The greatest success is the precision rate achieved from the 20 documents that were screened by ScHARR, showing that the system can successfully identify relevant documents for the ScHARRHUD and should be used on a large scale to aid in the retrieval of additional documents.

Chapter 7

Future work

7.1 Text mining accessibility: the production of a user-friendly tool for systematic reviewers

Throughout the development and research of this project, it became clear that there is potential for a re-usable, user-friendly tool that can implement text mining in the medical study area. Given the current scarcity of resources within the medical area, combined with the conclusions from the O'Mara study [5] that text mining should be considered safe for use in the ranking of documents for systematic review, the production of such a tool would be beneficial. With some small changes to the code produced for the project and the creation of a user interface, text mining could be made much more accessible for systematic reviewers.

The system would have a simple UI with three main pages: query page, test data page and a results page. The query page should include buttons for adding PMIDs to the query as well as importing a list from a file. The list of PMIDs added to the system from here should be written to the `query.txt` file already implemented by the system. The test data page should have features to download files from medline in a given year range or import test data files from a folder. The download from medline will run the `download_medline_files.py` script already written for this project with a small change to allow for the user to select range of medline files to be downloaded. The import from folder would set the path flag used in the core system script. Finally, the results page should simply provide a display for the results file created by the system, showing the ranking of the documents as well as their abstracts. Most of this functionality is already implemented by the system created for this project including the user customisability and so creation of this tool would be relatively simple. This would allow users to create their own queries whether it be for HSUVs or any other area in the medical field and run them against selected sections of the Medline database. This would make text mining more accessible for systematic reviewers, reducing workloads and improving the quality of results. Additional functionality could be added into this system for example, custom stop word lists, classifier evaluation (if a gold standard data set is provided) and options to turn on relevance feedback.

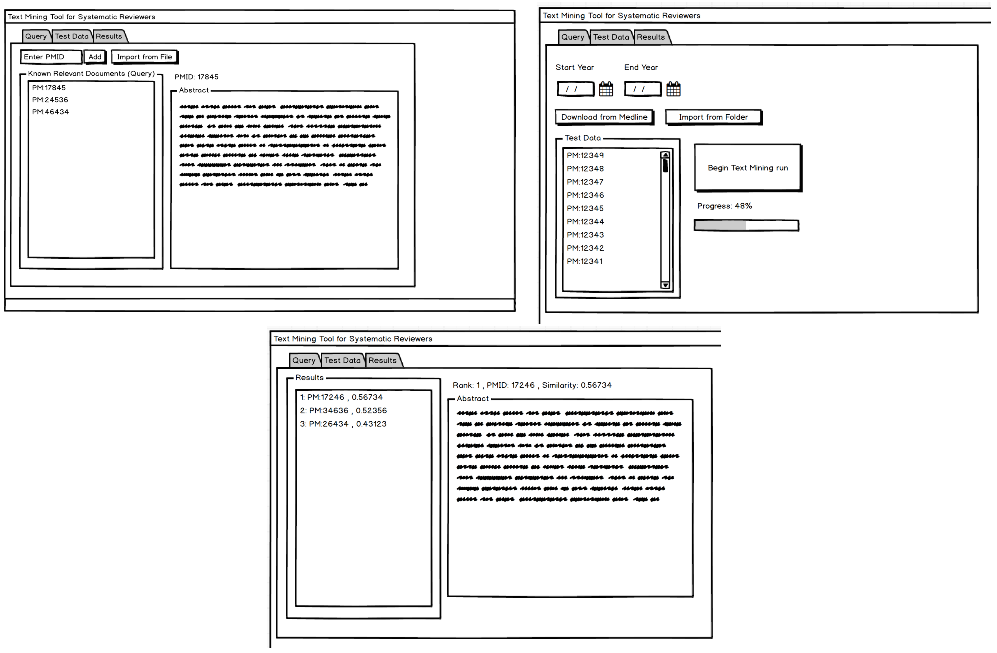


Figure 7.1: Mockups for the UI of a text mining tool for systematic reviewers

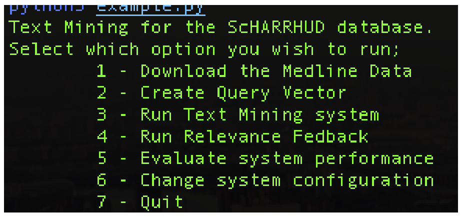
7.2 Evaluation of results by ScHARR

Due to a number of factors including time constraints and lack of available resources at ScHARR, only a small number of results were screened by experts at ScHARR. While evaluating the overall results of the system would have been beneficial to the project, it would require some dedicated resource by ScHARR. The upside of this however, is that the results from the project are not subject to change and will be available to ScHARR whenever they have the time to investigate. Additionally, the system will also be available to ScHARR allowing them to run further text mining experiments or updating the results when the latest Medline baseline repository becomes available.

7.3 Single file script

While designing the system, one of the objectives was to make everything available from a single script. However, once development began on the system, it became evident that there was too much functionality to put into a single script without overcomplicating it. In order to achieve all of the functionality behind a single script there were two possible options; creating a command line application or having a large number of flags to control every part of the system.

The command line application would involve running a main script which would provide the user with a starting menu from which they could select the operation they wish to perform. Once the operation had completed, it would return the user back to the menu meaning they would not be required to run scripts individually. This would have been an ideal solution and would be simple to change by using `import` to gain access to the functions in the individual files. Further development of the project in this way, could result in improved usability for system users.



```
python3 example.py
Text Mining for the ScHARRHUD database.
Select which option you wish to run;
        1 - Download the Medline Data
        2 - Create Query Vector
        3 - Run Text Mining system
        4 - Run Relevance Fedback
        5 - Evaluate system performance
        6 - Change system configuration
        7 - Quit
```

Figure 7.2: Example of the command line application

The second option to include a large number of flags to control the whole system from a single script would have overcomplicated the system and made it very clunky to use. Users would be required to construct long commands each time they wished to run the system and would require good knowledge of each flag and how to use it. As such, this is not considered to be a beneficial option for the system.

Chapter 8

Conclusions

Text mining applications are becoming more prominent in the medical field due to the large available dataset and the significant resultant reduction in required resources. The applications save reviewers a large amount of time screening irrelevant documents and present them with the most likely documents to be relevant to their desired area. ScHARR are currently undertaking a project to collect literature surrounding health utilities and store it in a database called ScHARRHUD. This project has created a text mining system to rank the available documents in the Medline Baseline repository in terms of their relevance to the literature already collected in the ScHARRHUD database.

The system was built with a focus on results, but with a secondary objective of being re-usable and customisable for future use. The system's performance during testing was very positive, achieving good base scores for the evaluation metrics. These results were further improved by the implementation of relevance feedback and the experiments to find the optimum values for each variable. Most notably, the system achieved a 75% precision rate when the top 20 results from a small testing sample were screened by experts at ScHARR proving the effectiveness of the system. This confirms the success of the project and confirms that it should be used on a large scale for further document identification.

Further developments to the system can be made to improve the results including further iterations of relevance judgements by experts at ScHARR being fed into the Rochhio algorithm. Additionally, there is the suggestion to develop a user interface for the system to improve accessibility to text mining in the medical field.

Bibliography

[1] National Institutes of Health, "Mesh browser website," *https://meshb.nlm.nih.gov/*, 2018.

[2] Suomela and Brian P and Andrade Miguel A, "Ranking the whole medline database according to a large training set using text indexing," *BMC bioinformatics*, vol. 6, no. 1, p. 75, 2005.

[3] M. Pai, M. McCulloch, J. D. Gorman, N. Pai, W. Enanoria, G. Kennedy, P. Tharyan, and J. J. Colford, "Systematic reviews and meta-analyses: an illustrated, step-by-step guide.," *The National medical journal of India*, vol. 17, no. 2, pp. 86–95, 2004.

[4] Z. Lu, "Pubmed and beyond: a survey of web tools for searching biomedical literature," *Database*, vol. 2011, 2011.

[5] C. Stansfield, A. O'Mara-Eves, and J. Thomas, "Reducing systematic review workload using text mining: opportunities and pitfalls," *Journal of the European Association for Health Information and Libraries*, vol. 11, no. 3, pp. 8–10, 2015.

[6] S. Ananiadou, B. Rea, N. Okazaki, R. Procter, and J. Thomas, "Supporting systematic reviews using text mining," *Social Science Computer Review*, vol. 27, no. 4, pp. 509–523, 2009.

[7] A. Rees, S. Paisley, J. Brazier, A. Cantrell, E. Poku, and K. Williams, "Development of the scharr hud (health utilities database)," *Value in Health*, vol. 16, no. 7, p. A580, 2013.

[8] D. Papaioannou, J. Brazier, and S. Paisley, "Systematic searching and selection of health state utility values from the literature," *Value in Health*, vol. 16, no. 4, pp. 686–695, 2013.

[9] H. Dakin, L. Abel, R. Burns, and Y. Yang, "Review and critical appraisal of studies mapping from quality of life or clinical measures to eq-5d: an online database and application of the maps statement," *Health and quality of life outcomes*, vol. 16, no. 1, p. 31, 2018.

[10] School of Health and Related Research: University of Sheffield, "Scharrhud," *https://Scharrhud.org*, 2018.

[11] G. Balestroni and G. Bertolotti, "Euroqol-5d (eq-5d): an instrument for measuring quality of life," *Monaldi Archives for Chest Disease*, vol. 78, no. 3, 2015.

[12] R. Talib, M. K. Hanif, S. Ayesha, and F. Fatima, "Text mining: Techniques, applications and issues," *International Journal of Advanced Computer Science & Applications*, vol. 1, no. 7, pp. 414–418, 2016.

[13] C. E. Lipscomb, "Medical subject headings (mesh)," *Bulletin of the Medical Library Association*, vol. 88, no. 3, p. 265, 2000.

[14] L.-P. Jing, H.-K. Huang, and H.-B. Shi, "Improved feature selection approach tfidf in text mining," in *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, vol. 2, pp. 944–946, IEEE, 2002.

[15] A. Huang, "Similarity measures for text document clustering," in *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pp. 49–56, 2008.

[16] P. Zweigenbaum, D. Demner-Fushman, H. Yu, and K. B. Cohen, "Frontiers of biomedical text mining: current progress," *Briefings in bioinformatics*, vol. 8, no. 5, pp. 358–375, 2007.

[17] A. M. Cohen and W. R. Hersh, "A survey of current work in biomedical text mining," *Briefings in bioinformatics*, vol. 6, no. 1, pp. 57–71, 2005.

[18] C. C. Aggarwal and C. Zhai, *Mining text data.* Springer Science & Business Media, 2012.

[19] J. Thomas, J. McNaught, and S. Ananiadou, "Applications of text mining within systematic reviews," *Research Synthesis Methods*, vol. 2, no. 1, pp. 1–14, 2011.

[20] C.-H. Wei, H.-Y. Kao, and Z. Lu, "Pubtator: a web-based text mining tool for assisting biocuration," *Nucleic Acids Research*, vol. 41, 07 2013.

[21] R. Jelier, M. Schuemie, A. Veldhoven, L. Dorssers, G. Jenster, and J. Kors, "Anni 2.0: a multipurpose text-mining tool for the life sciences.," 2008.

[22] A. OMara-Eves, J. Thomas, J. McNaught, M. Miwa, and S. Ananiadou, "Using text mining for study identification in systematic reviews: a systematic review of current approaches," January 2015.

[23] B. Settles, "Active learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, pp. 1–114, 2012.

[24] F. Sebastiani, "Machine learning in automated text categorization," *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp. 1–47, 2002.

[25] G. Salton and C. Buckley, "Improving retrieval performance by relevance feedback," *Journal of the American society for information science*, vol. 41, no. 4, pp. 288–297, 1990.

[26] T. Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization.," tech. rep., Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[27] E. Kanoulas, D. Li, L. Azzopardi, and R. Spijker, "Clef 2017 technologically assisted reviews in empirical medicine overview," in *CEUR Workshop Proceedings*, vol. 1866, pp. 1–29, 2017.

[28] A. Alharbi and M. Stevenson, "Ranking abstracts to identify relevant evidence for systematic reviews: The university of sheffield's approach to clef ehealth 2017 task 2: Working notes for clef 2017," in *CEUR Workshop Proceedings*, vol. 1866, CEUR, 2017.

[29] J. Thomas, "Text mining for reducing screening workload: is it safe to use?."

[30] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[31] A. R. Aronson, O. Bodenreider, H. F. Chang, S. M. Humphrey, J. G. Mork, S. J. Nelson, T. C. Rindflesch, and W. J. Wilbur, "The nlm indexing initiative.," in *Proceedings of the AMIA Symposium*, p. 17, American Medical Informatics Association, 2000.

[32] A. Bronshtein, “Train/test split and cross validation in python,” 2017.

[33] ranks.nl, “Long list of stop words produced by ranks.nl/stopwords.”

[34] PubMed, “Pubmed stopword list.”

[35] University of Glasgow Information Retrieval Group, “University of glasgow information retrieval group stop word list.”