IFN647 Advanced Information Retrieval and Storage
Project: EduSearch Information System

Total Marks: 100 (Scaled down to 40% of overall course mark)
Due Date: Monday 31st October 2016

Instructions: You should complete the project in teams of two or three (maximum). Only one student in each group is required to submit the assignment. However, all group members' names and student numbers need be included in the written report and all group members' student numbers also included in the final zip filename.

Section 1. Scenario

You have been hired by Kingsland University of Technology to implement a new information retrieval system (hereafter referred to as "the application") that will replace their existing system (hereafter referred to as the "baseline system"). The application should search and retrieve abstracts from academic publications based on queries. The application is to be implemented in C# and utilise the Lucene search engine library (Lucene.NET version 3.0.3). You are required to:

- Implement an information search and retrieval system in C# using Lucene.NET 3.0.3;
- Implement changes to the application to differentiate it from the baseline system. As part of these changes you ***must:*** (See Section 3 for more details)
    1. Pre-process the information need to create a query as the input into the application. Query creation must be automated and require no manual input from the user.
    2. Implement one or more **sensible and justifiable** changes to Lucene's "DefaultSimilarity Class"
- Evaluate the performance of the application and compare it to the baseline system;
- Produce a written report that describes, amongst other things, the design, testing, performance and instructions on how to use your application.

Important Points

- **For each of the information needs contained in "cran_information_needs.txt" file, the text contained in the description field *must* be pre-processed to create the query into your application. This processing must be automatic and require no manual alteration by the user.**
- **The source documents contain an error (the title is repeated in the abstract text). You need to handle this error as best you can.**
- **You may use screen shots of your application to prove that it is able to search and retrieve documents and screen shots of Luke.Net to show that your application is able to index the source documents.**
- **You should try to implement changes that improve the performance of your application with respect to the baseline system. However, you will not be penalised if your changes do not improve on the baseline as long as: your changes are reasonable and you have adequately evaluated and described the changes.**
- **Amongst other things, you must create a zip file containing all the source code associated with your application. This should include a visual studio project file that can be opened and built using Visual Studio 2015/Community on the computers in GP-S506. If the project is not able to be built using Visual Studio on the computers in GP-S506, then the associated functionality marks will be 0. (Note VS2015 may be read in this document *exactly* the same as Visual Studio Community)**

Section 2. Collection

You will be using the **Cranfield Test Collection** to evaluate your application. All files relating to the test collection are available on blackboard (**collection.zip**).

The collection consists of:

1. A set of source documents (**crandocs** directory);
2. A set of information needs (**cran_information_needs.txt**);
3. A set of relevance judgements (**cranqrel.txt**);

1. Source Documents

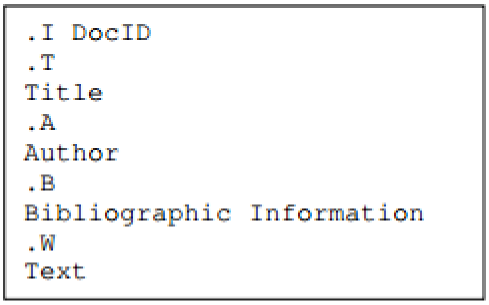
The source documents are a collection of journal article abstracts. Each document is saved as "DocID.txt" where DocID is a number from 1 to 1400 (1.txt to 1400.txt) that identifies the document. This DocID will also be used in the results file described in Section 3.

Each file contains the following information:

- DocID - the respective document id as discussed above;
- Title – the title of the article;
- Author – the author of the article;
- Bibliographic information - journal abbreviation, volume/number, year of publication and page number;
- Words - The abstract.

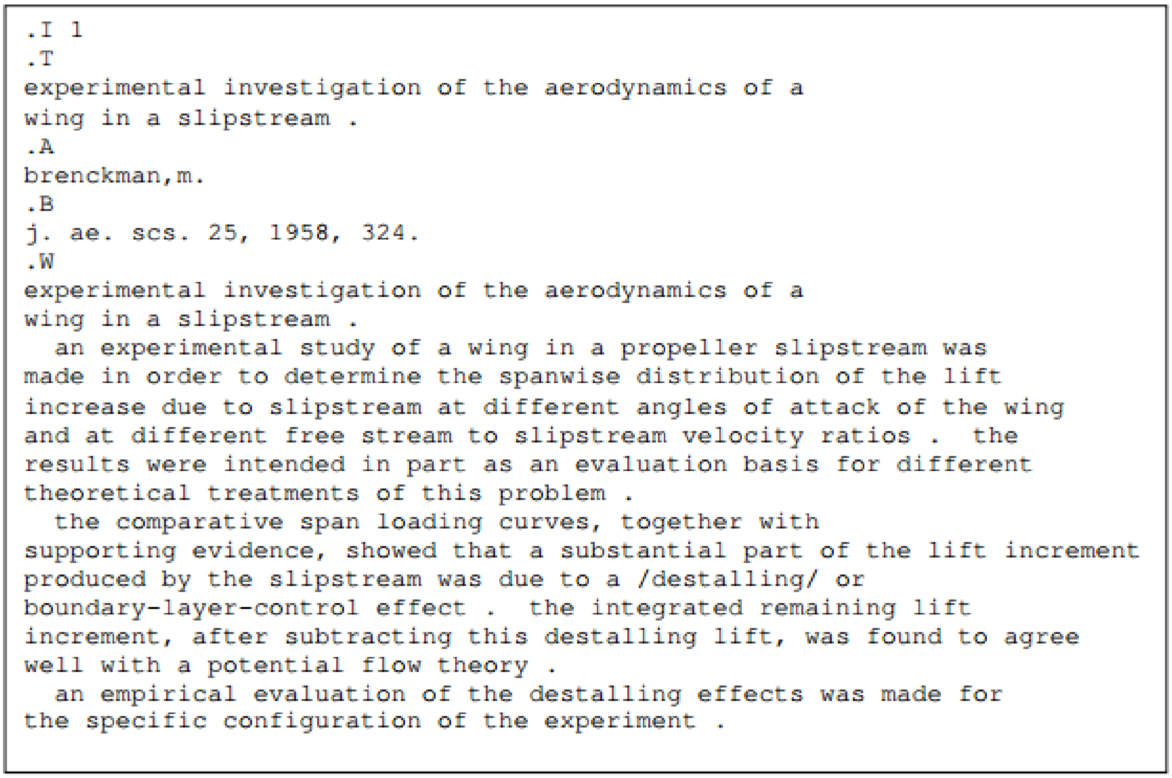
Each of these sections is specifically outlined in the source documents.

Certain delimiters (.I,.T,.A,.B and .W) are used to specify each section. The source documents are formatted as follows:



```
.I DocID
.T
Title
.A
Author
.B
Bibliographic Information
.W
Text
```

An example of a source document (1.txt) is presented below. Note that the title of the article is also included in the words of the abstract. **<u>This is an error which you need to deal with.</u>** How you chose to deal with this error in the document is an implementation choice.



```
.I 1
.T
experimental investigation of the aerodynamics of a
wing in a slipstream .
.A
brenckman,m.
.B
j. ae. scs. 25, 1958, 324.
.W
experimental investigation of the aerodynamics of a
wing in a slipstream .
  an experimental study of a wing in a propeller slipstream was
made in order to determine the spanwise distribution of the lift
increase due to slipstream at different angles of attack of the wing
and at different free stream to slipstream velocity ratios .  the
results were intended in part as an evaluation basis for different
theoretical treatments of this problem .
  the comparative span loading curves, together with
supporting evidence, showed that a substantial part of the lift increment
produced by the slipstream was due to a /destalling/ or
boundary-layer-control effect .  the integrated remaining lift
increment, after subtracting this destalling lift, was found to agree
well with a potential flow theory .
  an empirical evaluation of the destalling effects was made for
the specific configuration of the experiment .
```

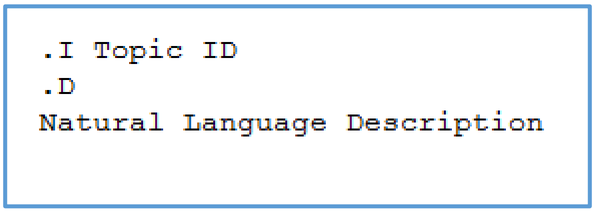
2. Information Needs

Each information need must be processed to formulate a query which will be used as the search input into your application. There are five information needs in total and contain the following information:

- TopicID - the respective topic id;
- Description – a full text description of the user's information need which is to be the basis for the query into the application. Your application will need to automatically process this information need to create the query which will be used as an input into your application. This is to be a completely automated process. You ***cannot*** use the entire description as the query input into your application.

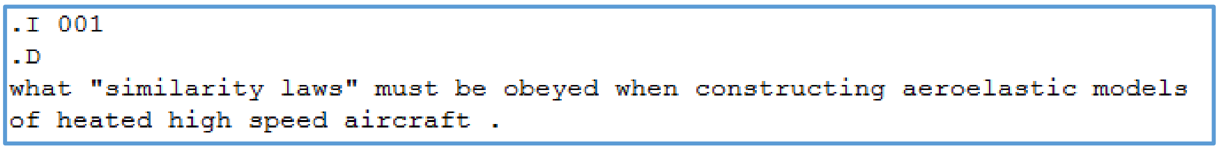
Each of these sections is specified in the information need file: **"cran_information_needs.txt"**

Certain delimiters (.I & D.) are used to specify each section. The information needs are formatted as follows:



```
.I Topic ID
.D
Natural Language Description
```

An example of an information need is presented below.



```
.I 001
.D
what "similarity laws" must be obeyed when constructing aeroelastic models
of heated high speed aircraft .
```

Note that the five (5) information need ids are: **001, 002, 023, 157 and 219.**

3. Relevance Judgements

The relevance judgements file outlines which of the documents in the source collection are relevant to each information need.

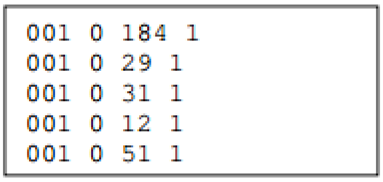
There is only one relevance judgement file. Each line of the file contains the following information:

- TopicID– The topic id number;
- 0 – The character 0;
- DocId - The document id of the relevant document;
- Judgment – A score of 1 indicating that the document is relevant

The relevance judgments are formatted as follows:

```
TopicID    0   DocId    Judgement
```

A sample of the relevance judgements file is provided below:



```
001  0  184  1
001  0  29  1
001  0  31  1
001  0  12  1
001  0  51  1
```

Section 3. System Requirements

You are required to produce an application written in C# that allows users to index, search and retrieve abstracts of journal articles based on a specific information need. It is assumed that you will produce a user friendly Graphical User Interface. The functionality of the application is broken up into the following tasks:

Task 1: Index

- On start-up your application should enable the user to specify two directory paths. The first directory path will contain the source collection. The other directory path will contain the location where a Lucene index will be built.
- The user will then be able to indicate that they would like to create the index from the specified collection (e.g. through a button click or similar) which is then done programmatically.
- The application should build the index and report the time it takes to index to the user.

Task 2: Search

- Once the index has been built the user must be able to search it.
- The user should be able to enter a natural language information need into a search box and the application must then process this information need to create a final query that is submitted into the application automatically
- The final query created programmatically from the natural language information need submitted by the user must be displayed on the GUI
- How the application processes the information need entered by the user is an implementation/design choice
- The user should have the option (via a checkbox/radio button or similar control) to submit multi-term and/or multi-phrase queries “as is” ***with no pre-processing***
- The application should match the final query to relevant documents and rank the documents according to relevance. The application should report how long it took to search the index and display this on the GUI (include the time required for query creation)
- The application should display in the GUI how many relevant documents were returned from the search
- The application should present a ranked list of the top 10 relevant documents. For each document the following pieces of information should be presented:
    - ➢ The title;
    - ➢ The author/s;
    - ➢ Bibliographic information (journal abbreviation, volume/number, year of publication, page number);
    - ➢ The first sentence of the abstract. Note: This does not mean the title (which is an error) but rather the first genuine line of the abstract. For example for document 1.txt this would be the line that starts “*an experimental study of a wing in…*”.

Task 3: Browse Results

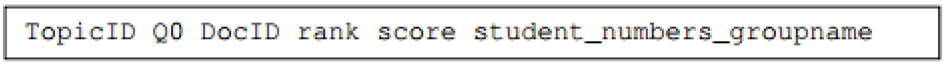
- After submitting their search the user should be able to browse the Top 10 results. The user should then be able to select the next 10 most relevant results (i.e. results ranked 11 – 20). The information required to be displayed is the same as for the Top 10 results.
- After browsing results 11-20 the user should be able to continue browsing the list of results. However, now the user can either move forward down the list (i.e. results ranked 21 – 30) or up the list (1-10).
- The user should be able to move backwards and forwards thorough the ranked list of results, viewing 10 results at a time.

Task 4: Retrieve Results

- Using an appropriate interface control, the user should be able view the entire abstract. This can either open in the existing window or a new window.

Task 5: Save Results

- The user must have the option of saving the list of the retrieved results in a text file. To do this the user will need to specify the name of the file to save the results and query identification. New results should be appended to the end of an existing results file.
- The format of the text file should be compatible with the *trec_eval* program. The format of which is as follows:

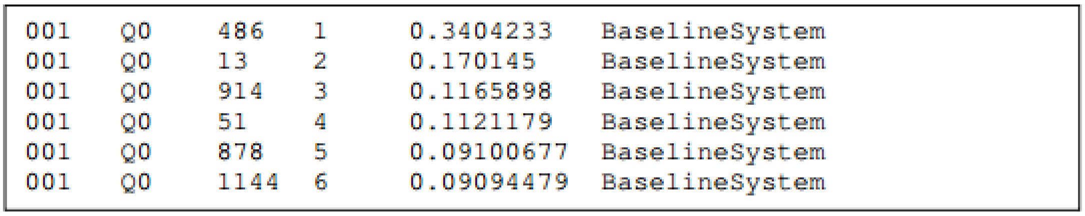


```
TopicID Q0 DocID rank score student_numbers_groupname
```

Where:

- ➢ TopicID is the query identification as entered by the user;
- ➢ Q0 is simply the two characters Q0;
- ➢ DocID is the respective document id;
- ➢ rank is the rank of the file as returned by your application;
- ➢ score is the relevancy score of the file as determined by your application;
- ➢ student_numbers_groupname are the QUT student numbers for all members of the group (delimited by underscores) and a group name (e.g 0123456798_0987654321_ourteam). This is used to identify the result file from your application.

All parameters are separated by whitespace. A sample of the results file from the BaselineSystem is provided next.



```
001   Q0   486    1    0.3404233    BaselineSystem
001   Q0   13     2    0.170145     BaselineSystem
001   Q0   914    3    0.1165898    BaselineSystem
001   Q0   51     4    0.1121179    BaselineSystem
001   Q0   878    5    0.09100677   BaselineSystem
001   Q0   1144   6    0.09094479   BaselineSystem
```

Note that in order to be compatible with *trec_eval* the results file needs to be a unix formatted file. You can use the program ***dos2unix*** to assist with this.

Task 6: Custom Similarity Modification

As part of the applications implementation you are required to make an improvement to the baseline scoring functionality of Lucene. In workshops and lectures you have been shown how Lucene calculates its scoring function. Lucene search is based on a Boolean and Vector Space Model with some minor modifications. Your application must make sensible and justifiable modifications to the Lucene scoring function by implementing a custom similarity class which inherits from the "DefaultSimilarity" base class.
http://lucenenet.apache.org/docs/3.0.3/d6/d95/class_lucene_1_1_net_1_1_search_1_1_default_similarity.html

Section 4. Report Requirements

In addition to the application you are also required to produce a report. The report should contain the following sections.

Part 1: Statement of Completeness and Work Distribution.

- The Statement of Completeness should be completed with respect to the functionality outlined in Section 3. If any item has not been completed then it should be detailed.
- The Work Distribution should list which pieces of work were completed by which team members. It should also how marks should be split between team members.
- The Statement of Completeness and Work Distribution needs to be signed by all members.

Part 2: Design

- A description of your overall system design, identifying which are the major classes and methods in your application and how they relate to the major components of an IR system (indexing, searching, human computer interface). A UML diagram may assist in this description.
- A description of your indexing strategy, including which classes and methods are responsible for indexing.
- A description of how you are handling the structure of the source document, including which parts of the source documents are being indexed or stored within the index and the reasons why you made these decisions.
- A description of how you are handling any document errors.
- A description of your search strategy, including which classes and methods are responsible for searching. Include your modifications to the "DefaultSimilarity" base class
- A description of your user interface, including interface mock-ups. You may use a number of applications to produce the mock-ups, however, ***you are not allowed to use screen shots of your final application***. ***You need to provide mock-ups for all of the functionality described in Section 3 to show your GUI design process.***

Part 3: Changes to Baseline

- A list of all the changes that you have made in comparison with the baseline system, with a short description (1-2 sentences) of the change.
- Describe the change(s) made to the "DefaultSimilarity" class and the reason for the change(s)
- For ***two (2)*** of the changes listed above a longer description on:
    - The motivation behind the change and why you think the change will lead to improved search performance;
    - Where in your application the changes have been implemented.
- You should try to implement changes that improve the performance of your application with respect to the baseline system. ***However, you will not be penalised if your changes do not improve on the baseline as long as your changes are reasonable and you have adequately evaluated and described the changes.***

Part 4: System Evaluation

- Provide a suitable test plan and subsequent results for the system evaluation presented in Section 6. You may use screen shots of your application to prove that it is able to search and retrieve documents and screen shots of ***Luke.Net*** to show that your application is able to index the collection.

Part 5: Comparison with Baseline

- Provide a comparison with the baseline system as indicated using the criteria in Section 7.

Part 6: User Guide

- Provide a user guide that instructs how to perform each of the tasks outlined in Section 3.

Part 7: Advanced Features

- Identify two "advanced features" either identified from the material covered in class or from your own research.
- For each of the advanced features provide:
    - A description of the advanced feature;
    - An explanation of why you think it would aid in retrieval performance;
    - A description of what changes would be required in your application to implement the advanced features and where in your application (i.e. class/methods) you would need to make them.

NOTE: For these advanced features you are not required to implement them but rather describe and justify why they would improve the application.

Section 5. Baseline System

The baseline system has also been developed in C# and Lucene.Net. The properties of the baseline system are as follows:

- For each file in the collection, all of the text is indexed as a single field.
- The Analyzer used during index and search is the **Lucene.Net.Analysis.Simple analyzer**.
- The index does not save information related to field normalisation.
- The index does not save information related to term vectors.

Section 6. System Evaluation

Your application needs to be evaluated both in terms of **efficiency** and **effectiveness**.

The required **efficiency metrics** are:

- **Index size;**
- **Time to produce index;**
- **Time to search.**

The required efficiency metrics should be provided across the source documents (for the index size and time to produce the index) and across all five test information needs (for the time to search). The time must include the time for processing the information need to produce the final query into the application. You may use ***Luke.Net*** to find the size of the index.

The required **effectiveness metrics** are:

- **MAP;**
- **Overall precision;**
- **Overall recall;**
- **Precision at 10.**

The required effectiveness metrics should be calculated using the provided five test natural language information needs contained in the "**cran_information_needs.txt"** file. Results should be generated for each information need individually and then across all five information needs. The program ***trec_eval*** can be used to assist you in calculating these metrics. Depending on how you have formatted your results file, you may also need to use the program ***dos2unix***.

Section 7. Comparison with Baseline System

You should compare the performance of your application with the baseline. This should be done in five ways:

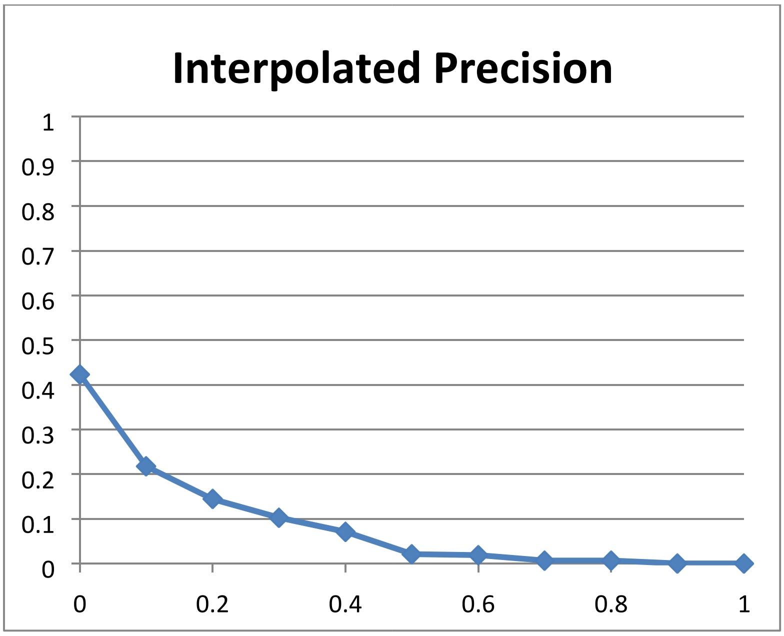
1. A comparison of the performance of your application against the baseline system using the three efficiency metrics outlined in Section 6.
2. A comparison of the performance of your application against the baseline system using the four effectiveness metrics outlined Section 6 for each of the five test natural language information needs and overall.
3. A graph that shows the overall Interpolated Recall-Precision Averages at 11 standard values, between your application and the baseline system for all of the five test information needs.
4. A significance test that compares the reported MAP values of your application with the baseline system across the five test information needs. You should use the 1 tail, paired, t-test to perform this analysis and state whether your system is significantly better than the baseline at significance level of 0.05. $(p \leq .05)$ (*See Workshop 6)*
5. A short description that describes the reasons why you believe your system outperforms or does not outperform the baseline system.

The results for the Baseline system are as follows:

Metric	**Value**
Index Size	4166 kb
Index Time	0.362 (s)
Search Time	0.092 (s)

Topic	MAP	Precision	Recall	Precision@10
1	0.2247	0.0985	0.4483	0.4000
2	0.1028	0.0355	0.6800	0.1000
23	0.0422	0.0318	0.8788	0.0000
157	0.0028	0.0201	0.1250	0.0000
219	0.0020	0.0113	0.1579	0.0000
All	0.0749	0.0329	0.4589	0.1000

Recall	Interpolated Precision
0	0.4232
0.1	0.2175
0.2	0.1445
0.3	0.1027
0.4	0.0709
0.5	0.0211
0.6	0.0188
0.7	0.0066
0.8	0.0066
0.9	0
1	0



Section 8. Late submission and plagiarism policies

QUT’s standard policy for late submissions and plagiarism will apply. Information on these policies can be found at the following

- https://www.student.qut.edu.au/studying/assessment/late-assignments-and-extensions
- http://www.mopp.qut.edu.au/C/C_05_03.jsp
- http://www.mopp.qut.edu.au/E/E_08_01.jsp#E_08_01.08.mdoc

Section 9. What to Submit

By the due date you need to submit:

- A zip file containing all the source code associated with your application. This should include a Visual Studio project file that can be opened and built using Visual Studio 2015/Community on the computers in GP-S506. If the project is not able to be built using Visual Studio 2015/Community on the computers in GP-S506, then the associated functionality marks will be 0.
- A PDF or word document of your report. Ensure that all group member’s names and student numbers are on the report. The report should be well presented and formatted and use appropriate headings where necessary. The report should use be professional in nature.
- A results file generated from your application as specified in Section 3. The results file should be able to be executed by the ***trec_eval*** program on the computers in GP-S506.

Section 10. Assessment Criteria

The assessment criteria are separated into two parts:

1. **System functionality**
2. **Written report.**

Marks will be assigned as follows. Please note that marks may be deducted for failing to meet criteria, poor communication, inadequate descriptions or other reasons.

Functionality Assessment Criteria

Part	**Marks**
System with Baseline Functionality	15
Option to submit query "as is"	5
Pre-process Natural Language Query	5
Implement changes to Lucene Scoring	5
Implement two changes to Baseline System	10

Report Assessment Criteria

Part	**Marks**
Statement of Completeness	2
Design	10
Changes to Baseline	10
System Evaluation	15
Comparison to Baseline	10
User Guide	5
Advanced Features	8
Report Total	**60**

Total Marks

Part	**Marks**
Functionality	40
Report	60
Report Total	**100**

<< END OF PROJECT SPECIFICATION >>