A User Guide for Flex

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Bhagyashri A. Mahule

In Partial Fulfillment of the requirements

for the degree of

Master of Science

Dr. Carolyn J. Crouch

August, 2011

Acknowledgements

I sincerely thank my adviser Dr. Carolyn Crouch for her invaluable support throughout my graduate career, for her guidance and the sharing of her knowledge and understanding of the Information Retrieval field. I also owe thanks to Dr. Chris Price, Dr. Ted Pedersen, Dr. Hudson Turner, Dr. Pete Willemsen, Dr. Jim Allert for all their help and support in past two years.

I specially acknowledge Dr. Joe Gallian for being available on a short notice for our defense. I would also like to thank him for providing me a good mathematical foundation which helped me understand many Computer Science concepts. I would also like to extend thanks to Dr. Ryan Rosandich for teaching me *Advanced Project Management* subject which would definitely be helpful for me to succeed in my professional career.

I would like to specially thank Radhika Banhatti, Natasha Acquilla and Reena Rachel for being excellent team mates to work with. I would also like to thank Sai Subramanyam Chittala and Supraja Nagalla for their help in processing the collection. I also acknowledge special thanks to Dinesh Bhirud, Murthy for giving us useful response inspite of their busy professional schedule.

I would also like to thank the Department of Computer Science at University of Minnesota Duluth. I would specially acknowledge Clare Ford and Jim Luttinen for their assistance. Also, special thanks to Lori for lot of things (list will never end).

Special thanks to my husband, Abhijeet Mahule for his unconditional love and support throughout my life. I would like to thank my family and all my friends for their support throughout my academic career and personal life. I would like to convey my special thanks to all my classmates for always being cheerful. You all made my stay in Duluth enjoyable.

Abstract

Information retrieval focuses on organizing and storing data and retrieving useful information from it. With the advent of the web, the XML has became a preferred standard for storing documents. The documents which are available on the Web can be structured as well as semi-structured.

Traditionally, information retrieval was concentrating on retrieving the entire document that is relevant to the query. But with XML retrieval, the elements of the document at various levels of granularity can be retrieved instead of the document as a whole. We call this type of retrieval *flexible retrieval*.

As the software process matures, the code expands and becomes more difficult to debug and maintain. Having good documentation is critical. This year, we extend the work done by previous teams. Flex has been streamlined and applied to a newly parsed collection which gave us better, cleaner text. The main focus of this thesis is to document the complex process of flexible retrieval.

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

Information retrieval involves organizing and storing data and selecting useful information from it. The world of the Web or Internet is constantly increasing as is the information in it. The information available on the Web is vast, and it is critical to extract useful content from these available resources. This is why the companies like Yahoo, Google and Microsoft are continuously working on improvements in the field of information retrieval (IR).

Documents available on Web can be *structured, unstructured or semi-structured.* Structured documents follow all Document Type Definition (DTD) constraints. These documents are well-defined. This means that all text lies within XML tags, and so each portion of the document is identifiable. Unstructured documents lack all these constraints and semi-structured documents contain at least some untagged text.

A traditional IR systems retrieves, for each query, a set of documents which correlate with it. There are various models to support this task. Perhaps the most common model is Salton's Vector Space Model [12]. In this model, both documents and queries are converted into vectors; and these vectors are weighted by term frequency. The correlated documents are found by applying a similarity measure such as cosine or inner product.

INEX (INitiative for Evaluation of XML documents) is a competition which promotes Web-based (i.e., XML) retrieval. The University of Minnesota Duluth participates in this competition every year. INEX provides each participant a set of documents, evaluation packages and metrics by which results of competing teams can be compared.

In 2005, major progress was made by the UMD IR group with the advent of the Flex retrieval system. This system enables us to perform dynamic element retrieval. Much of the implementation was done by Khanna [10]. In 2006, the INEX document collection changed to Wikipedia. A major contribution was made by Ganapathibhotla [4] who derived a methodology for computing the values of the global parameter n_k (in the *ltu-query* weighting formula) at execution time, without access to a global dictionary.

Chapter 2 describes background including Vector Space Model [12], the Smart retrieval engine [13] and the environment required by INEX for element retrieval. Chapter 3 is a user manual for Flex. It describes all steps to be followed (from scrubbing the collection to INEX evaluation) in producing results for INEX 2009-10 Ad hoc tasks.

2 Background

This chapter presents a brief overview of the Vector Space Model [12], the basic model in Information Retrieval, Smart [13] retrieval engine. It gives a brief description of the document and topic collections, the INEX 2009 and 2010 tasks, relevance assessment, and the evaluation measures used.

2.1 Retrieval Model

Vector Space Model

The basic retrieval model used in our research is the Vector Space Model [12]. In this model, each document and query is represented as an *n*-dimensional vector, where n is the number of terms in the vector. Each term in the vector represents a unique word in the document or query. These terms are weighted by their frequencies. A similarity measure, such as cosine or inner product, is used to measure similarity.

2.2 Smart Retrieval Engine

We use Smart [13] as the retrieval engine in our work. This engine is based on the Vector Space Model. A Smart retrieval produces a ranked list of articles or elements for each query. Smart provides basic functionalities, such as indexing, term weighting, and retrieval.

2.3 INEX

INEX [6] is an INitiative in XML Information Retrieval. It facilitates the development of different strategies to improve the retrieval of XML elements. (Web-based systems focus on retrieving to the granularity of sections, subsections, passages and even snippets). A current aim of INEX is the evaluation of focused retrieval. For this purpose, every year INEX provides participants with a large collection of documents and a sample set of queries to be used with it. Relevance assessments are performed by participants, who evaluate the queries using their search engines and then submit their results to INEX. These results are compared using different metrics and the results are published. INEX includes a number tracks such as Ad Hoc, Book, Question Answering, etc. The University of Minnesota Duluth participates in the Ad Hoc track every year.

2.4 Wikipedia Document Collection

The document collection used for this research is the 2009 Wikipedia XML document collection [8] provided by INEX. It is composed of a large number of English articles covering very diverse topics. This collection, 50.7 GB in size, contains 2,666,190 Wikipedia articles having more than 30,000 unique tags. (It can be downloaded in four parts from the INEX website and is divided into 1000 directories). This collection of XML documents does not strictly follow a DTD (Document Type Definition). The documents are semi-structured; i.e., untagged text is present in them. A sample Wikipedia document is shown in Figure [1].

2.5 Topic Collection

Queries are in the form of CO + S (Content Only + Structure) [9]. There are 107 queries contained 2010 collection. A typical query is seen in Figure [2]. We use only the *title* field for our queries.

2.6 Relevance Assessments

Relevance assessment is the manual assessment of a set of documents for a query. Each participant group is provided with a subset of documents and queries. The participant reads each document and assesses whether that document is relevant to a particular query or not. The assessments are then pooled to form the set of all documents judged relevant to the query. This set forms the basis for evaluating the participants' results. The assessments from all participants are collected to form a relevance assessment pool. Each year INEX provides a tool (conventionally *GPXRai)* which is used to evaluate the text. INEX converts this marked text into File Offset and Length (FOL) format which can be used for further evaluation. File offset represents the best entry point in each document and length represents the number of characters relevant to the topic. The qrels file which contains all this information is provided by INEX for use in evaluation.

```
<?xml version="1.0" encoding="UTF-8"?>
<article>
<name id="1516">
Albert the Degenerate
</name>
<conversionwarning>
0
</conversionwarning>
<body>
<template name="Merge">
Albrecht II, Markgraf of Meißen
</template>
<emph3>
Albert, surnamed "The Degenerate"
</emph3>
<unknownlink src="13 November">
13 November
…</article>
```

Figure 1: Sample Document, Document Id: 1516.xml (2008)

2.7 Retrieval Tasks

The INEX 2009 Ad hoc track includes three major retrieval tasks [5], namely, Thorough, Focused and Relevant in Context (RiC).

INEX 2009 Tasks

Thorough Task

The primary goal of this task is to return a ranked list of relevant elements grouped by document. It may contain overlapping elements. (e.g., if the child element is relevant then its parent is also relevant, and both are returned.) For evaluating this task, *MAiP*, i.e., Mean Average Interpolation Precision, is used. See [5].

```
<inex_topic query_type="CO+S">
<title>Indian food</title>
<castitle>//article[about(., Indian food)]</castitle>
<phrasetitle>" Indian food "</phrasetitle>
<description>Get information about Indian food.</description>
<narrative> India is a vast country with varying cultures, languages,
climates and people. Each region in India has its specific food specialty.
Cooking is my hobby and I love Indian food. I always try to cook new
Indian dishes.
Following results can be considered relevant. Anything that is related to
Indian food, anything related to famous hotels and their favorite dishes.
Anything related to various spices, herbs, vegetables, fruits that are
grown in India. Anything related to breakfast, lunch, snacks,0 beverages
from India.Following results can be considered irrelevant. The
documents which is not related to Indian food anyway. Or the document
that’s just mentions Indian food but does not contain any information
related to it.
</narrative>
</inex_topic>
```

Figure 2: Sample Query, Query Id: 2010003 (2010)

Focused Task

The primary goal of this task is to return a ranked list of non-overlapping (i.e., focused) relevant elements. In terms of the XML tree, if a child element is relevant to the query, then its parent is also relevant; the most relevant element must be returned. So this task focuses on identifying the best choice among overlapping elements. For evaluation of this task, Interpolation Precision (iP) is used. See [5].

Relevant In Context (RiC) Task

The primary goal of this task is to return a ranked list of non-overlapping elements, in rank order by a document. In this task, the top-ranked documents are selected and then all focused elements from each document is returned. For evaluating the RiC task, Mean Average Generalized Precision (*MAgP)* is used. See [5].

INEX 2010 Tasks

The tasks performed in 2010 are different versions of the 2009 tasks with more focus on the length of the retrieved element. A small portion of the document called a *snippet* is retrieved. These new tasks are described below. See [2].

Efficiency Task

This task is same as the 2009 Thorough task except that more focus is given to the trade-off between efficiency and effectiveness. For each topic, a ranked list of possibly overlapping elements (15, 150, 1500) is retrieved. The participants are supposed to report I/O costs and execution times for these runs on a query basis. The hardware and software requirements of the machines are also considered. The effectiveness-efficiency metric is formed using these inputs. For evaluating this task, MAiP, i.e., Mean Average Interpolation Precision, is used. See [2].

Restricted Focused Task

This task is a variant of the 2009 Focused task. (The summary of each document having length up to 1000 characters is retrieved. This means that output is restricted to 1000 characters per query). For evaluation of this task, Interpolation Precision (iP) metric is used. See [2].

Relevant In Context (RiC) Task

This task is identical to the 2009 RiC task. For evaluating the RiC task, Mean Average Generalized Precision*, or MAgP,* is used. See [2].

Restricted Relevant In Context (RRiC) Task

The 2010 task is identical to 2009 task except that in 2010, the length of the result is limited. Only 500 characters which per topic are retrieved. The same *MAgP* metric, is used to evaluate this task. See [2].

3 USER MANUAL

This chapter describes all steps to be followed (from scrubbing the collection till INEX evaluation) in producing results for INEX 2009-10 Ad hoc tasks. The collection provided by INEX is in raw format. The procedure required to make that collection suitable for Smart to do indexing and retrieval is explained in section *3.1* and *3.2*. Next section, 3.*3* describes the steps to index the document and topic collection. The process of article retrieval using Smart [13] is explained in the section *3*.4 and Flexible retrieval is explained in the section *3.5*. In the end, the last section *3.6* describes the procedure to evaluate the Flex retrieval for various INEX tasks, such as Thorough, Focused, Restricted Focused, Relevant in Context, and Restricted Relevant in Context.

3.1 Scrubbing :

The collection given by the INEX is in raw format (i.e., it contains non-needed tags). To clean these XML documents the process of scrubbing is done. The steps carried out during this process are:

1. To scrub all documents a shell script ***sample.sh*** is used. It takes ***/smart/2010_collection/tar_collection/PAGES/pages*** (our raw document collection) as input directory and cleans each document from it and copies the new document to the output directory ***/smart/chit0051/parses***.

    Location : ***/smart/chit0051/sample.sh***
    Usage *: ./sample.sh Input_dir_to_raw_collection*

output_dir_to_save_scrubbed_collection

<u>*Input*</u> *: Input folder where raw collection is present*

<u>*Output*</u> *: Output folder to save scrubbed collection*

e.g., ./scrub /smart/2009_collection/raw_collection /smart/2009_collection/scrubbed_collection

This script uses following scripts to do scrubbing.

i. Placing tags in raw XML file – The script ***basic1.pl*** is used for placing all tags in the raw xml file in a new line.

<u>*Location*</u> : ***/smart/chit0051/basic1.pl***

<u>*Usage*</u> *: perl /smart/chit0051/basic1.pl originalfile.xml > rev-xmlfile.xml*

<u>*Input*</u> *: Raw document*

<u>*Output*</u> *: Modified document with non-terminal tags only*

ii. To remove non-alphanumeric characters the script, ***basic2.pl*** is used.

<u>*Location*</u> : ***/smart/chit0051/basic2.pl***

<u>*Usage*</u> *: perl basic1.pl rev-xmlfile.xml > rev1-xmlfile.xml*

<u>*Input*</u> *: Output of previous step*

<u>*Output*</u> *: Modified document with no punctuations*

e.g., perl basic1.pl rev-xmlfile.xml > rev1-xmlfile.xml

iii. The following program scrub.c does two things :

- Removal of the tags with “word-net” attribute and the non-needed tags like *id, revision, link*
- It also places un-tagged text into *mt* tags.

Location : ***/smart/chit0051/scrub.c***

Usage : ./scrub rev1-xmlfile.xml > rev2-xmlfile.xml

Input : Output of previous step

Output : Scrubbed document

e.g., ./scrub rev1-xmlfile.xml (output of basic2.pl) > rev2-xmlfile.xml

iv. ***scrub1.c*** appends all the *mts* in all given levels at that level.

Location : ***/smart/chit0051/scrub1.c***

Usage : ./scrub rev2-xmlfile.xml > rev3-xmlfile.xml

Input : Output of previous step

Output : Scrubbed document with mts

e.g., ./scrub rev2-xmlfile.xml > rev3-xmlfile.xml

Now the collection is scrubbed.

3.2 Parsing :

Parsing Document Collection :

The scrubbed documents are parsed at different levels. The process of parsing is done through following steps:

1. To get maximum level of parsing a document : ***maxlevel.c***

Location : ***/smart/chit0051/scripts/maxlevel.c***

Usage : ./maxlevel filename.xml

Input : Input xml file

e.g., ./maxlevel filename.xml

2. The following two steps are executed through a shell script ***para1.sh***

<u>Location</u> : ***/smart/chit0051/scripts/para1.sh***

<u>Usage</u> : sh para1.sh

i. The output of above script is given to another program ***prog5.c*** which does parsing starting at max level upto level-0.

<u>Location</u> : ***/smart/chit0051/scripts/prog5.c***

<u>Usage</u> : ./prog5 xml_file_name.xml levelcount>filename_level_count_parse.txt

<u>Input</u> : xml_file_name.xml - Input xml file to be parsed

levelcount – level upto which the parsing is to be done

<u>Output</u> : filename_level_count_parse.txt - Parsed output file

e.g., ./prog5 xml_file_name.xml levelcount > filename_level_count_parse.txt

Output of this process is all parses (level-0 to max-level).

ii. This step removes additional new line characters and spaces from the parses of all the levels

<u>Location</u> : ***/smart/chit0051/scripts/basic3.pl***

<u>Usage</u> : .perl /smart/chit0051/scripts/basic3.pl filename_level_count_parse.txt > parsed_file.txt

<u>Input</u> : filename_level_count_parse.txt – Output of previous step

<u>Output</u> : parsed_file.txt – Cleaned output file

e.g., perl /smart/chit0051/scripts/basic3.pl filename_level_count_parse.txt > parsed_file.txt

3. The script ***para2.sh*** does following:

a. Creates para-parse
b. Created para+mt parse
c. Creates doctrees for para-parse and para+mt parse

Location : ***/smart/chit0051/scripts/para2.c***

Usage : sh para2.sh

> Para and para+mt parse is done using ***prog3.c***

Location : ***/smart/chit0051/scripts/prog3.c***

Usage : ./prog3 xml_file_name.xml para > filename_para_parse.txt

Input : xml_file_name.xml - Input xml file to be parsed

para or mt – para for para parse and mt for para+mt parse

Output : filename_para_parse.txt - Parsed output file

e.g., ./prog3 xml_file_name.xml para > filename_para_parse.txt

> The script ***preorder.c*** is used to generate doctrees for the whole collection.

Location : ***/smart/chit0051/scripts/preorder.c***

Usage : ./preoder xml_file_name.xml>xml_file_name.tree

Input : xml_file_name.xml - Input xml to create doctree

Output : xml_file_name.tree – Doctree of input file

e.g., ./preoder xml_file_name.xml > xml_file_name.tree

4. The extended doctrees are created using ***preorder1.c.*** Scrubbed document is traversed in pre-order and the result is a doctree.

Location : ***/smart/chit0051/scripts/preorder1.c***

<u>Usage</u> : ./preoder1 xml_file_name.xml (original xml file)
>xml_file_name.tree
<u>Input</u> : xml_file_name.xml - Input xml to create doctree
<u>Output</u> : xml_file_name.tree – Doctree of input file
e.g., ./preoder1 xml_file_name.xml > xml_file_name.tree

Now all document processing is complete.

<u>Parsing Query Collection</u> :
INEX provides the query collection in the form of an xml file. We parse this file and then index it to convert it to *doc.Lnu* vector. Following steps are executed to do parsing.

1. This step splits the query file into individual files having one query per file. It also removes extra tags from query file. It just keeps *article*, *body* and *fno* tags (required by indexing script) and removes all other tags.
Script used : ***para_parserNT.pl***
<u>Location</u> : /smart/09_scripts/parsing_scripts/para_parserNT.pl
<u>Usage</u> : perl para_parserNT.pl config_file path_to_query_collection
path_to_output log_file
<u>Input</u> : config_file – Config file specifying tag to keep and tags to index
path_to_query_collection – Location of folder where query collection is stored
log_file – Log file path
<u>Output</u> : path_to_output – Location to create output
e.g., perl /smart/09_scripts/parsing_scripts/para_parserNT.pl /smart/09_scripts/parsing_scripts/nconfig_queries.txt /smart/IR2009-2011/2009_Queries/2009-topics.xml /smart/IR2009-2011/2009_Queries/ParsingQueries/2009_Queries/ /smart/tmp/logfile

The config file looks like :

```
>cat /smart/09_scripts/parsing_scripts/nconfig_queries.txt
        tags_to_keep
        title
        tags_to_index
        title
```

The text of the query is enclosed in <*title*> - </*title*> tag. The *title* tag must be used in two places in the config file because the same script is used for document, element and query indexing.

2. We again concatenate these query files into a single file. We just append the queries sequentially.

Script used : ***concatenateAllQueries.pl***

<u>Location</u> : /smart/10_scrpts/concatenateAllQueries.pl

<u>Usage</u> : perl Input_Directory (input parsed queries) Outputfile

<u>Input</u> : Input_Directory – Location of output directory of previous step

<u>Output</u> : Outputfile – File to concatenate all queries

e.g., perl /smart/10_scrpts/concatenateAllQueries.pl /smart/IR2009-2011/2009_Queries/ParsingQueries/2009_Queries/ /smart/IR2009-2011/2009_Queries/ParsingQueries/concatenatedQueries.txt

3. Queries can contain +, - and " characters as well as words. We remove all these characters, retaining the terms associated with + and " character.

Script used : ***cleanQueries.pl***

Location : /smart/10_scrpts/cleanQueries.pl

Usage : cleanQueries.pl concatenatedQueries_file clean_parsed_queries_file

Input : concatenatedQueries_file – output file of previous step

Output : clean_parsed_queries_file – Clean file with "+" and "-" removed

e.g., perl /smart/10_scrpts/cleanQueries .pl /smart/IR2009-2011/2009_Queries/ParsingQueries/concatenatedQueries.txt /smart/IR2009-2011/2009_Queries/ParsingQueries/cleanedQueries.txt

3.3 Indexing :

The Smart retrieval engine is used for indexing. It converts each element into *nnn* vector as per the term frequency. If we give article parse an as input to Smart for indexing then it converts each article into *nnn* vector and for element parse it creates *nnn* vector per element.

Procedure :

To index any parsed elements -

Scripts :

generate_docloc.pl – *Creates doc-loc file which contains xpaths and their physical location (disk address)*

generate_docid_docpath_mapping.pl – *Creates a file which has mappings between Smart and Wiki ids*

Config files :

The other files that are required are :

inex_query_loc* , *spec.parse_inex *(contains tags required for indexing)***,**
make_inex_index_nnn_documents.

All these files can be found at ***: /smart/09_scripts/indexing_scripts/***

Procedure Required for each level of Indexing

Directory structure need to be created manually (for each level 0-7)

- Create a folder for level_0 indexing, e.g., *indexing_level_0*
- Copy spec.parse_inex file into this folder
- Use generate_docloc.pl script to generate doc-loc file.

Pictorial view :

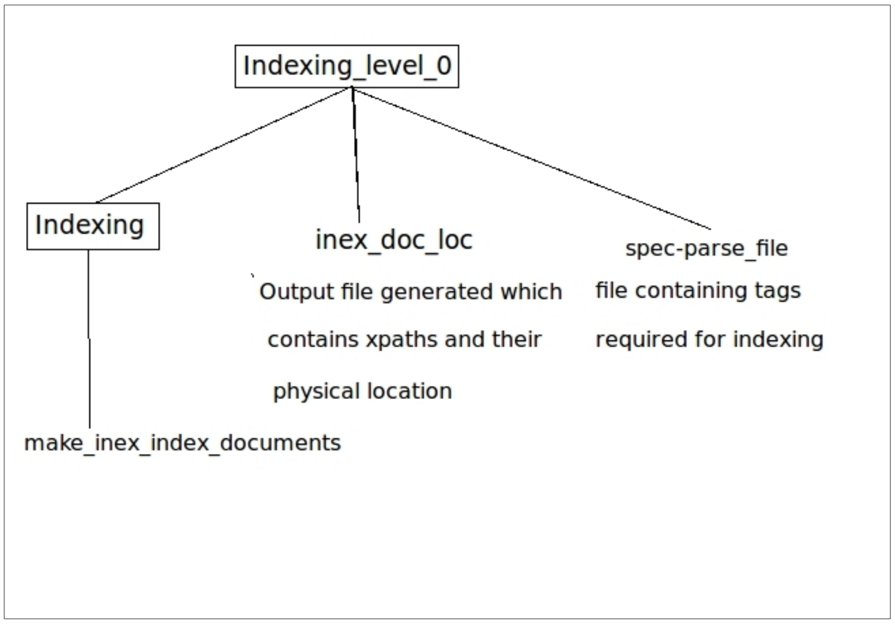
Indexing_level_0
Indexing
inex_doc_loc
spec-parse_file
Output file generated which contains xpaths and their physical location
file containing tags required for indexing
make_inex_index_documents

Figure 3: Pictorial view of directory structure for Indexing (for level_0 indexing)

Usage of ***generate_docloc.pl:***

perl generate_docloc.pl absolute_path_to_parsed_collection absolute_path_to_index_folder_inex_doc_loc_file

<u>Location</u> *: /smart/09_scripts/indexing_scripts/generate_docloc.pl*

<u>Input</u> *: Absolute path of parsed collection*

<u>Ouput</u> *: Absolte path of output*

e.g., perl generate_docloc.pl /smart/2011_Collection/parsed_articles
/smart/2011_Collection/Indexing_level_0/inex_doc_loc -- >
creates *inex_doc_loc* file for *articles or level_0.*

Note: Always give absolute path (path from root) to the collection. This is really important or else indexing will not be done.

- Create another folder and name it ***indexed*** (Please do not use any other name for this folder).
- After the ***indexed*** folder is created copy ***make_inex_index_nnn_documents*** into it.
- After above step we have the ***inex_doc_loc*** and ***spec-parse-file*** files and a folder named ***indexed*** in the folder *indexing_level_0* for level_0.
- Open the ***make_inex_index_nnn_documents*** file and change the ***$current*** to the path of the folder where the ***indexed*** folder is created. In this case ***$current*** is set to

/smart/2011_Collection/Indexing/indexing_level_0

- Run the script ***make_inex_index_nnn_documents***

  e.g., ./make_inex_index_nnn_documents

Now indexing process for level_0 is completed.

Repeat the same procedure for all levels (***level_1 to level_7***) and ***para*** and ***para+mt*** indexing.

Flex uses level_0 and para+mt *nnn* vectors only.

To do all element indexing -

1. Create a folder *all_element_index folder* and Copy the *spec.parse_inex* file
2. Copy the ***inex_doc_file*** of ***level_0*** into the ***all_element_index*** folder.

The next thing to be done is the ***inex_doc_file*** of all levels (***level_0 to level_7***) and ***para*** should be appended to the ***inex_doc_file*** of ***level_0***, so the single ***inex_doc_file*** has the path of elements of all levels**.**

– Copying the ***inex_doc_loc*** file

e.g., cp /smart/2011_Collection/Indexing_level_0/inex_doc_loc /smart/2011_Collection/Indexing_all_element_new/inex_doc_loc

– Create ***inex_doc_loc*** file for *all_element_index*

>cd /smart/2011_Collection/Indexing_level_0

> more inex_doc_loc >> /smart/2011_Collection/Indexing_all_elements_new/inex_doc_loc

Do above two steps for all levels, from 0 to 7 and *para.*

This concatenates the *inex_doc-loc* files for all levels into a single file (required for all- element retrieval)

After going through the above mentioned steps we will have a ***inex_doc_loc*** file in the ***all_element_index*** that contain path of all elements at all levels.

3. Now the ***all_element_index*** folder has ***inex_doc_loc*** file, ***spec.parse_inex*** file and the ***indexed*** folder. The ***indexed*** folder has ***make_inex_index_nnn_documents*** file. Go into the ***indexed*** folder.

e.g., cd /smart/2011_Collection/Indexing/all_element_index_new/indexed

- Run the script ***make_inex_index_nnn_documents***

e.g., ./make_inex_index_nnn_documents

4. *Now we have to use the* ***generate_docid_docpath_mapping.pl*** *script to create a mapping between Smart and Wiki ids of all documents. This script uses* ***textloc.txt*** *created to create all_el_docid_docpath.txt.* The text loc file has the actual (physical) location of each parsed element.

<u>*Location*</u> : /smart/09_scripts/indexing_scripts/
generate_docid_docpath_mapping.pl
<u>*Usage*</u> : *perl generate_docid_docpath_mapping.pl*
/smart/2011_Collection/Indexng_all_elements_new/indexed/tex
tloc
/smart/2011_Collection/Indexing_all_elements_new/indexed/
all_el_docid_docpath.txt.
<u>*Input*</u> *: Location of textloc file generated during indexing*
<u>*Output*</u> *: Dociddocpath mappig file*

To Split docid-docpath-mapping file into folders -

Now we have docid-docpath-path-mappings generated for all levels in one file. We have to split this in number of different files for each article. This step is very important because seeding of doctrees will not take place without this. For efficiency, the *docid-docpath-mapping.txt* is split into different folders.

<u>*Script used*</u> : ***split_docid_docpath_mapping.pl***
<u>*Location*</u> *: /smart/10_scrpts/split_docid_docpath_mapping.pl*
<u>*Usage*</u> *: perl Input_file Outputfolder*
<u>*Input*</u> *: Input_file – Location of docid-docpath-mappings file created (of Paras+mt)*
<u>*Output*</u> *: Outputfile – Folder to split this file into different files*

e.g., perl split_docid_docpath_mapping.pl

/smart/2011_Collection/Indexing/Indexing_parasMT/indexed/docid_docpath_mapping.txt

/smart/2011_Collection/DocIDDocPath_parasMT/

This step splits all-element docid-docpath file into 1000 sub-folders. Now we have indexed all-element collection.

To index queries for all-element retrieval (NOT Flex!) :

The query.*nnn* is identical for article and all-element index.

Steps to be followed while indexing are :

- Create a folder ***query_indexing*** where you want to index the queries.
Then copy ***inex_query_loc, spec.parse_inex*** into the above folders. They are in /smart/09_scripts/indexing_scripts/

- Create another folder and name it ***indexed*** (Please do not use any other name for this folder). After the ***indexed*** folder is created copy ***make_inex_index_nnn_queries*** into the ***indexed*** folder.

- Use the ***generate_queryloc.pl*** script to create the ***inex_query_loc*** file. The ***inex_query_loc*** file contains path to each query.

Usage of ***generate_queryloc.pl:***

perl generate_queryloc.pl absolute_path_to_parsed_collection absolute_path_to_index_folder_inex_doc_loc_file

Location : /smart/09_scripts/indexing_scripts/generate_queryloc.pl

Input : Absolute path of parsed queries

Ouput : Absolte path of output

e.g., perl generate_queryloc.pl /smart/IR2009-
2011/2009_Queries/ParsingQueries/2009_Queries/
/smart/2011_Collection/Indexing_queries/inex_query_loc -- >
creates *inex_doc_loc* file for queries.

Note: Always give ***absolute path*** *(path from root)* to the collection. This is really important or else indexing will not be done.

- Then run *make_inex_index_nnn_queries*
nohup ./make_inex_index_nnn_queries

Indexing creates number of files which are not in readable (text) format. Smart has given us a utility, *smprint* to convert these files into text format if desired.

File conversion commands:

3. *cd /smart/2011_Collection/Indexing_all_elements_new/indexed*
4. */smart/smart.13.0/src/bin/smprint dict dict.words > dict.words.txt*
5. */smart/smart.13.0/src/bin/smprint vec doc.nnn > doc.nnn.txt*
6. */smart/smart.13.0/src/bin/smprint textloc textloc > textloc.txt*
7. */smart/smart.13.0/src/bin/smprint vec query.nnn > query.nnn.txt*
8. */smart/smart.13.0/src/bin/smprint inv inv.words > inv.words.txt*

3.4 Article Retrieval using Smart :

After indexing the collection and queries next step is to retrieve the top *n* documents using Smart. We give document vectors and query vectors to Smart engine. There are number of weighting schemes which we can apply through Smart. For our purpose, we use the *Lnu-ltu* weighitng scheme. Smart converts *doc.nnn* to *doc.Lnu* and *query.nnn* to *query.ltu*. Smart uses the *inner product* similarity measure for calculating similarity. Before applying a similarity measure Smart converts theses vectors to normalized form based on the *Lnu-ltu* weighting formula.

Article retrieval i1s done in two steps :

1. <u>Term weighting for articles and queries</u> :

In this step, *doc.nnn* vectors are converted to *doc.Lnu and query.nnn* vectors are converted to *query.ltu* vectors. This step uses a config file, namely, ***nnn_to_Lnu.***

Modify following parameters in this file:

$current – location of indexing folder of article (level_0)
e.g., set current = /smart/2011_Collection/Indexing/Indexing_level_0

$database – location where you want to create database (output of this step). The database folder has *doc.Lnu* and *query.ltu* files.
e.g., set database = /smart/IR2009-2011/slope_pivot_exp_articles/s0.05_p156/large_set/database

$slope – Current Slope value for article retrieval
e.g., slope 0.04

$pivot – Current Pivot value for articles retrieval
e.g., pivot 156

<u>Location</u> : */smart/09_scripts/other_scripts/nnn_to_Lnu*
<u>Usage</u> : *nohup ./nnn_to_Lnu*

2. <u>Smart Retrieval for articles</u> :

In this step the actual retrieval of articles (or elements is) done. The ***retrieve_Lnu*** is the config file used in this step. This step creates a list of top *n* documents, ***tr.Lnu.txt.***

$database – location where you have created database folder in previous step.
e.g., set database = /smart/IR2009-2011/slope_pivot_exp_articles/s0.05_p156/large_set/database

$Lnu_retrieval – location of retrieval (output) folder. Smart retrieves articles and put that list into this folder.
e.g., set Lnu_retrieval =/smart/IR2009-2011/slope_pivot_exp_articles/s0.05_p156/large_set/Lnu_retrieval

$num_wanted – Number of articles to be retrieved
e.g., num_wanted 1500

<u>Location</u> : /smart/09_scripts/other_scripts/retrieve_Lnu
<u>Usage</u> : nohup ./retrieve_Lnu

The output file produced by Smart is not ordered by correlation. To order the output file by correlation use following command.

```
sort -n -k1,1 -k3,3 Lnu_retrieval/tr.Lnu.txt > sorted_tr.Lnu.txt
```
output – rank-ordered list of articles

3.5 Flex Operations :

1. <u>For Query Weighting for use in Flex</u> *(query.nnn* to *query.ltu* using all-element Slope and Pivot*):*

This step is done for weighting query vector, *query.nnn* to *query.ltu* using all-element Slope and Pivot.

$current – location of indexing folder of elements (all_element)

e.g., set current =

/smart/2011_Collection/Indexing/Indexing_all_elements_new

$database – location where you want to create database (output of this step).

The database folder has *doc.Lnu* and *query.ltu* files.

e.g., set database = /smart/IR2009-2011/slope_pivot_exp_articles/s0.05_p156/large_set/database

$slope – All-element Slope value

e.g., slope 0.11

$pivot – All-element Pivot value

e.g., pivot 38

<u>Location</u> : /smart/09_scripts/other_scripts/nnn_to_Lnu

<u>Usage</u> : nohup ./nnn_to_Lnu

Now we have an *ltu*-weithted query for use by Flex.

2. <u>Create a Subset of doctrees</u> :

The raw doctrees for whole collection are created during parsing step. We need to identify the *n* doctrees as a subset for the *n* documents retrieved in article retrieval.

i. <u>Create article list from (sorted) tr.Lnu.txt</u>

Smart engine gives the list of articles with Smart ids. We need to to convert this list of articles to a format which contains Wiki ids. The article list is created by using ***tr.Lnu.txt*** and ***docid_docpath_mapping.txt*** of article index. The perl script used for this step is : ***get_article_list.pl*** *(translates Smart ids to Wiki ids)*

<u>Location</u> : /smart/10_scrpts/get_article_list.pl
<u>Usage</u> : perl /smart/10_scrpts/get_article_list.pl
location_of_docid_docpath_mapping_for_articles
location_of_tr.Lnu.txt output_file_name
<u>Inputs</u> : location_of_docid_docpath_mapping_for_articles (level_0)
location_of_tr.Lnu.txt
<u>Outputs</u> : Output_file_name (article list)

e.g., perl /smart/10_scrpts/get_article_list.pl
/smart/2011_Collection/Indexing/Indexing_level_0/indexed/doc
id_docpath_mappings.txt Lnu_retrieval/tr.Lnu.txt
article_list.txt

ii. <u>Extract the doctrees for this article list</u>

Now we have the article list and all raw doctrees. We will extract a doctree for each of top *n* articles in this list using the perl script : ***copying_seeding.pl***

Location : /smart/10_scrpts/copying_seeding.pl

Usage : perl /smart/10_scrpts/copying_seeding.pl article_list location_of_raw_doctrees output_folder_to_create_doctrees_for_the_article_list

Inputs : article_list – Got from Smart retrieval or Reference list given by INEX

location_of_raw_doctrees – contains raw trees for entire collection

Outputs : output_folder_to_create_doctrees_for_the_article_list – output folder to copy the doctrees only for the list of articles in article list

e.g., perl /smart/10_scrpts/copying_seeding_files.pl article_list.txt /smart/2011_Collection/doctrees/ raw_trees

3. Seed these extracted doctrees :

To seed these doctrees we use : ***seeddoctrees.pl***

Location : /smart/10_scrpts/seeddoctrees.pl

Usage : perl seeddoctrees.pl docid-docpathfolder (of paras_mt) inputfolder (raw doctrees of the articles we need) outputfolder (seeded trees)

NOTE : Please make sure that the output folder does not exist!! while running this

Inputs : docid-docpath-folder – Folder where doc-doc-paths of para_mt are stored

location_of_raw_doctrees – doctrees only for the list of articles in article list (output of previous step)

Outputs *: output_folder – output folder to copy seeded doctrees*

e.g., perl /smart/10_scrpts/seeddoctrees.pl
/smart/2011_Collection/DocIDDocPath_parasMT/ raw_trees/
seeded_doctrees

The output of this step is seeded doctrees for top *n* articles.

4. Flex Retrieval :

Flex takes the seeded doctrees, nnn vectors for para+mt termonal nodes, *query.ltu*, and top *n* article list as inputs. Flex ouputs a ranked ordered list of elements as per their correlation with each query. It also builds the tree structure for each document.

The config file used for Flex is ***config_flex.txt.*** *Change following parameters in it.*

$DOC_INDEX_PATH– location of indexed parasMT collection (*doc.nnn*)

e.g., DOC_INDEX_PATH
/smart/2011_Collection/Indexing/Indexing_parasMT/indexed/doc.nnn

$*ARTICLE_LIST* – location of article list (Use reference list given by INEX or the once got from Smart)

e.g., ARTICLE_LIST /smart/IR2009-2011/slope_pivot_exp_articles/s0.05_p156/large_set/best_article_list.txt

Use this with pre-calculated *ltu* weighted queries.

$ QUERY_LTU_PATH – location of ltu-weighed query with all-element values of Slope and Pivot

e.g., QUERY_LTU_PATH
/smart/SlopePivot2011Elements/Queries_2010/database_0.11_38/query.ltu.txt

NOTE : We get this file during Conversion step for all elements (Once we finalize the slope, use that query.ltu.txt file)

Use this with Muthy's method.

$QUERY_INDEX_PATH - location of indexed query collection (*query.nnn*) (for use in Murthy's method)

e.g., QUERY_INDEX_PATH /smart/2011_Collection/2010Queries/allElement_queries/indexed/query.nnn

$OUTPUT_PATH – Location and name of the output file

e.g., OUTPUT_PATH /smart/IR2009-2011/slope_pivot_exp_articles/s0.05_p156/large_set/flex_out.txt

$RESULT_TREES_PATH – Location of seeded doctrees

e.g., RESULT_TREES_PATH /smart/IR2009-2011/slope_pivot_exp_articles/s0.05_p156/large_set/seeded_doctrees

$SLOPE_ALLELEMS – Slope value for all element retrieval

e.g., SLOPE_ALLELEMS 0.11 (for 2010)

$ PIVOT_ALLELEMS - Pivot value for all element retrieval

e.g., PIVOT_ALLELEMS 38 (for 2010)

Location : /smart/10_scrpts/config_flex.txt

Usage : nohup /smart/Flex/FlexDriver config_flex.txt

Produces for each query, *n* complete (i.e., populated) doctrees, parent elements are generated, and and all elements are *Lnu-weighted* correlating with the (*lnu-weighted*) query. The output is a list of XML elements, in rank order, in document order.

3.6 : Evaluation of Retrieval

<u>For Thorough task (2009 and 2010)</u>:

1. <u>Removing mts from Flex output</u>

The output of Flex contains some mts. To remove these mts, use the script, ***remove_mt_from_flex_output.pl***

> *<u>Location</u> : /smart/10_scrpts/remove_mt_from_flex_output.pl*
> *<u>Usage</u> : perl /smart/10_scrpts/remove_mt_from_flex_output.pl*
> *flex_output_file flex_output_with_no_mts*
> *<u>Inputs</u> : flex_output_file – Output of Flex*
> *<u>Outputs</u> : flex_output_with_no_mts – Output of this script*
>
> *e.g., perl /smart/10_scrpts/remove_mt_from_flex_output.pl flex_out.txt*
> *no_mt_flex.txt*

2. <u>Converting Flex output to INEX form</u>

The output of Flex is a ranked list of elements. To evaluate it using INEX, it should be converted to INEX format using the script : ***convert_to_inex_v2007.pl***.

> *<u>Location</u> : /smart/10_scrpts/convert_to_inex_v2007.pl*
> *<u>Usage</u> : perl /smart/10_scrpts/convert_to_inex_v2007.pl 72 thorough 1*
> *element 80000 2010001 no_mt_flex.txt flex_inex_format.xml*
> *<u>Inputs</u> : 72 – Participant id*
> *thorough – Task id*
> *1 – Task id (e.g., UMD_FOCUSED)*
> *element – Identifier for evaluation of all element retrieval*
> *80000 – Max. number of elements for all queries*

2010001 – First query id

no_mt_flex.txt – Flex-output without mts (output of previous step)

Outputs *: flex_inex_format.xml – Flex output converted to INEX format*

e.g., perl /smart/10_scrpts/convert_to_inex_v2007.pl 72 thorough 1 element 80000 2010001 no_mt_flex.txt flex_inex_format.xml

3. Converting Flex output to TREC format

The output of above step generates output in xml format. The INEX eval tool needs the output in TREC format. It has xpaths for all elements and corresponding document and element ids. To convert this into TREC format use the script : ***XMLtoTRECformat.pl***

Location *: /smart/10_scrpts/XMLtoTRECformat .pl*

Usage *: perl /smart/10_scrpts/XMLtoTRECformat .pl flex_inex_format.xml 1 80000 trec_format.txt*

Inputs *: flex_inex_format.xml – Flex output converted to INEX format (output of previous step)*

1 – Task id (e.g., UMD_FOCUSED)

80000 - Max. number of elements for all queries

Outputs *: trec_format.txt – Output TREC format*

e.g., perl /smart/10_scrpts/XMLtoTRECformat.pl flex_inex_format.xml 1 80000 trec_format.txt

4. Patching the xpaths

The previous step output has xpaths of all elements containing the tags which are not scrubbed off, e.g., article, bdy, sec, para etc. To convert these xpaths to ones with all intermediate tags use the script : ***ExpandXpathsNewMethod.pl***

<u>Location</u> : */smart/10_scrpts/ExpandXpathsNewMethod.pl*

<u>Usage</u> : *perl /smart/10_scrpts/XMLtoTRECformat .pl trec_format.txt expandedFile.txt extended_doctrees_folder*

<u>Inputs</u> : *trec_format.txt – TREC format output (output of previous step)*

extended_doctrees_folder – location of extended doctrees for entire collection

<u>Outputs</u> : *expandedFile.txt – Output file having xpaths comtaining all tags*

e.g., perl /smart/10_scrpts/ExpandXpathsNewMethod.pl trec_format.txt expandedFile.txt /smart/chit0051/doctrees_extended1

5. <u>Convert xpaths into FOL formats</u> :

Convert expanded xpaths to FOL formats using INEX SUBTOFOLeconomy jar.

<u>Location</u> : */smart/IR2010/2010_assessment_tool/2010-assessment-tool/SUB2FOLeconomy.jar (2010)*

/smart/IR2010/2009_assessment_tool/assessment-tool/2009-sub2fol-all/sub2fol/SUB2FOL.jar (2009)

<u>Usage</u> : *java -Xmx16G -jar location-of-SUB2FOLeconomy.jar expandedFile.txt SubtoFOL.trec++ location-of-gpxrai.table*

<u>Inputs</u> : *expandedFile.txt – File having xpaths with all tags (output of previous step)*

location-of-gpxrai.table – Location of gpxrai table given by INEX

<u>Outputs</u> : *SubtoFOL.trec++ – Output in FOL format*

Example command for 2009 task :

e.g., java -Xmx16G -jar /smart/IR2010/2009_assessment_tool/assessment-tool/2009-sub2fol-all/sub2fol/SUB2FOL.jar s_expandedFile.xml *sub2FOL.txt /smart/IR2010/2009_assessment_tool/assessment-tool/2009-sub2fol-all/sub2fol/gpxrai50rel.table.serialized*

Example command for 2010 task :

e.g., java -Xmx16G -jar /smart/IR2010/2010_assessment_tool/2010-assessment-tool/SUB2FOLeconomy.jar expandedFile.txt SubtoFOL.txt /smart/IR2010/gpxrai.table

6. Removing negatives from FOL :

Now in modified Flex we are not specifying the number of elements to be retrieved. Instead, Flex retrieves all relevant elements and we chop most relevant elements from it. There are few elements for which has correlation value as negative in FOL. We remove these elements and push the elements up to fill the window. The script used for this is : ***FOLRemoveNegatives.pl***

Location : /smart/10_scrpts/FOLRemoveNegatives.pl

Usage : perl /smart/10_scrpts/FOLRemoveNegatives.pl 1500 SubtoFOL.trec++ 1500_SubtoFOL.txt

Inputs : 1500 – Length of window, i.e., number of elements per query

SubtoFOL.txt – Output in FOL format

Outputs : 1500_SubtoFOL.txt – Output file with 1500 top elements per query

e.g., perl /smart/10_scrpts/FOLRemoveNegatives.pl 1500 SubtoFOL.trec++ 1500_SubtoFOL.txt

7. Use the INEX tool for evaluation :

We use the tool provided by INEX for evaluating our runs.

<u>Location</u> : smart/IR2010/2010_assessment_tool/2010-assessment-tool/inex_eval.jar

<u>Usage</u> : java -jar /smart/IR2010/2010_assessment_tool/2010-assessment-tool/inex_eval.jar -t qrels_file 1500_SubtoFOL.txt > MAIPvalue.txt

<u>Inputs</u> : -t – For Thorough task

qrels_file – Location of qrels file given by INEX

1500_SubtoFOL.txt – Chopped SubTOFOL file

<u>Outputs</u> : MAIPvalue.txt - It prints the output on cmd which can be redirected to this file

Example command for 2009 task :

```
java -jar
/smart/IR2010/2009_assessment_tool/inex_eval/inex_eval/inex_eval.jar -t
/smart/IR2010/2009_assessment_tool/inex_eval/inex_eval/inex2009.qrels
s_1500_SubtoFOL.txt > MAIP.txt
```

Example command for 2010 task :

```
e.g., java -jar /smart/IR2010/2010_assessment_tool/2010-assessment-
tool/inex_eval.jar -t   /smart/IR2010/2010_assessment_tool/2010-assessment-
tool/inex2010.qrels 1500_SubtoFOL.txt      > MAIPvalue.txt
```

<u>For Focused task</u> :

1. <u>Removing mts from Flex output</u> :

The output of Flex contains some mts. To remove these mts, use the script, ***remove_mt_from_flex_output.pl***

<u>Location</u> : /smart/10_scrpts/remove_mt_from_flex_output.pl

<u>Usage</u> : perl /smart/10_scrpts/remove_mt_from_flex_output.pl

flex_output_file flex_output_with_no_mts

<u>*Inputs*</u> *: flex_output_file – Output of Flex*

<u>*Outputs*</u> *: flex_output_with_no_mts – Output of this script*

e.g., perl /smart/10_scrpts/remove_mt_from_flex_output.pl flex_out.txt no_mt_flex.txt

2. <u>Removing overlaps</u> :

For focused evaluation, the overlaps are not allowed. The output of Flex is a ranked list of elements which might contain some overlaps. This step selects an element with higher correlation.

<u>*Location*</u> *:* /smart/10_scrpts/generate_focused_correlation_strategy.pl

<u>*Usage*</u> *:* /smart/10_scrpts/generate_focused_correlation_strategy.pl *no_mt_flex.txt* output.txt

<u>*Inputs*</u> *: no_mt_flex.txt – Flex-output with no mts (Output of step 1 in evaluation)*

<u>*Outputs*</u> *: output.txt – Output file generated after removing overlaps*

Example command for correlation strategy :

e.g., perl /smart/10_scrpts/generate_focused_correlation_strategy.pl no_mt_flex.txt correlation_strategy.txt

Example command for section strategy :

e.g., perl /smart/10_scrpts/generate_focused_section_strategy.pl no_mt_flex.txt section_strategy.txt

Example command for child strategy :

e.g., perl /smart/10_scrpts/generate_focused_child_strategy.pl no_mt_flex.txt child_strategy.txt

3. Converting Flex output to INEX form :

The output of Flex is a ranked list of elements. To evaluate it using INEX, it should be converted to INEX format using the script : ***convert_to_inex_v2007.pl***.

Location : /smart/10_scrpts/convert_to_inex_v2007.pl

Usage : perl /smart/10_scrpts/convert_to_inex_v2007.pl 72 focused Focused element 80000 2010001 section_strategy.txt flex_inex_format.xml

Inputs : 72 – Participant id

focused – Task id

Focused – Task id (e.g., UMD_FOCUSED)

element – Identifier for evaluation of all element retrieval

80000 – Max. number of elements for all queries

2010001 – First query id

no_mt_flex.txt – Flex-output without mts (output of previous step)

Outputs : flex_inex_format.xml – Flex output converted to INEX format

e.g., perl /smart/10_scrpts/convert_to_inex_v2007.pl 72 focused Focused element 80000 2010001 section_strategy.txt flex_inex_format.xml

4. Converting Flex output to TREC format :

The output of above step generates output in xml format. The INEX eval tool needs the output in TREC format. It has xpaths for all elements and corresponding document and element ids. To convert this into TREC format use the script : ***XMLtoTRECformat.pl***

Location : /smart/10_scrpts/XMLtoTRECformat .pl

Usage : perl /smart/10_scrpts/XMLtoTRECformat .pl flex_inex_format.xml 1

80000 trec_format.txt

<u>*Inputs*</u> *: flex_inex_format.xml – Flex output converted to INEX format (output of previous step)*

1 – Task id (e.g., UMD_FOCUSED)

80000 - Max. number of elements for all queries

<u>*Outputs*</u> *: trec_format.txt – Output TREC format*

e.g., perl /smart/10_scrpts/XMLtoTRECformat.pl flex_inex_format.xml 1 80000 trec_format.txt

5. <u>Patching the xpaths</u> :

The previous step output has xpaths of all elements containing the tags which are not scrubbed off, e.g., article, bdy, sec, para etc. To convert these xpaths to ones with all intermediate tags use the script : ***ExpandXpathsNewMethod.pl***

<u>*Location*</u> *: /smart/10_scrpts/ExpandXpathsNewMethod.pl*

<u>*Usage*</u> *: perl /smart/10_scrpts/XMLtoTRECformat .pl trec_format.txt expandedFile.txt extended_doctrees_folder*

<u>*Inputs*</u> *:   trec_format.txt – TREC format output (output of previous step)*

extended_doctrees_folder – location of extended doctrees for entire collection

<u>*Outputs*</u> *: expandedFile.txt – Output file having xpaths comtaining all tags*

e.g., perl /smart/10_scrpts/ExpandXpathsNewMethod.pl trec_format.txt expandedFile.txt /smart/chit0051/doctrees_extended1

All steps are common for 2009 and 2010 Focused task till here.

6. Convert xpaths into FOL formats :

Convert expanded xpaths to FOL formats using INEX SUBTOFOLeconomy jar.

Example command for 2009 task :

e.g., java -Xmx16G -jar /smart/IR2010/2009_assessment_tool/assessment-tool/2009-sub2fol-all/sub2fol/SUB2FOL.jar s_expandedFile.xml sub2FOL.txt /smart/IR2010/2009_assessment_tool/assessment-tool/2009-sub2fol-all/sub2fol/gpxrai50rel.table.serialized

Location : */smart/IR2010/2009_assessment_tool/assessment-tool/2009-sub2fol-all/sub2fol/SUB2FOL.jar*

Usage : */smart/IR2010/2009_assessment_tool/assessment-tool/2009-sub2fol-all/sub2fol/SUB2FOL.jar expandedFile.txt SubtoFOL.txt location-of-gpxrai.table*

Inputs : *expandedFile.txt – File having xpaths with all tags (output of previous step)*

location-of-gpxrai.table – Location of gpxrai table given by INEX

Outputs : *SubtoFOL.txt – Output in FOL format*

Example command for 2010 task :

e.g., java -Xmx16G -jar /smart/IR2010/2010_assessment_tool/2010-assessment-tool/SUB2FOLeconomy.jar expandedFile.txt SubtoFOL.txt /smart/IR2010/gpxrai.table

Location : */smart/IR2010/2010_assessment_tool/2010-assessment-tool/SUB2FOLeconomy.jar*

Usage : *java -Xmx16G -jar location-of-SUB2FOLeconomy.jar expandedFile.txt SubtoFOL.trec++ location-of-gpxrai.table*

Inputs : *expandedFile.txt – File having xpaths with all tags (output of* *previous step)*

location-of-gpxrai.table – Location of gpxrai table given by INEX

Outputs *: SubtoFOL.trec++ – Output in FOL format*

7. Snippet retrieval step (for 2010 task) :

This step is carried out for 2010 task only. It's for snippet retrieval. It chops the output to 1000 characters per element

Script used : ***snippet_retrieval.pl***

Location *:* /smart/10_scrpts/*snippet_retrieval.pl*

Usage *:* /smart/10_scrpts/ *snippet_retrieval.pl SubtoFOL.trec++ snippet_SubtoFOL.trec++*

Inputs *: SubtoFOL.trec++ – The file in SubtoFOL format (Output of step 6 in evaluation)*

Outputs : *snippet_SubtoFOL.trec++ – Output file generated with snippets*

e.g., perl /smart/10_scrpts/snippet_retrieval.pl SubtoFOL.trec++ snippet_SubtoFOL.trec++

8. Removing negatives from FOL (for 2009 and 2010 task) :

Now, in modified Flex we are not specifying the number of elements to be retrieved. Instead, Flex retrieves all relevant elements and we chop most relevant elements from it. There are few elements for which has correlation value as negative in FOL. We remove these elements and push the elements up to fill the window. The script used for this is : ***FOLRemoveNegatives.pl***

Location *: /smart/10_scrpts/FOLRemoveNegatives.pl*

Usage *: perl /smart/10_scrpts/FOLRemoveNegatives.pl 1500 SubtoFOL.trec++ 1500_SubtoFOL.txt*

Inputs *: 1500 – Length of window, i.e., number of elements per query*

SubtoFOL.txt – Output in FOL format

Outputs : *1500_SubtoFOL.txt – Output file with 1500 top elements per query*

e.g., perl /smart/10_scrpts/FOLRemoveNegatives.pl 1500 SubtoFOL.trec++ 1500_SubtoFOL.txt

9. Restricting the output to 1000 characters (for 2010 task) :

This step is carried out for Restricted Focused (2010) task only. It chops the output to 1000 characters per query.

Script used : ***get_focused_Restricted.pl***

Location : /smart/10_scrpts/*get_focused_Restricted.pl*

Usage : /smart/10_scrpts/ *snippet_retrieval.pl subtoFOL_50.trec++ subtoFOL_1000.trec++*

Inputs : *subtoFOL_50.trec++ – The output of previous step having 50 elements per topic*

Outputs : *subtoFOL_1000.trec++ – Output file generated having 1000 characters per topic*

e.g., perl /smart/10_scrpts/get_focused_Restricted.pl subtoFOL_50.trec++ subtoFOL_1000.trec++

10. Use the INEX tool for evaluation (for 2009 and 2010 task) :

Location : */smart/IR2010/2010_assessment_tool/2010-assessment-tool/inex_eval.jar (2010)*

/smart/IR2010/2009_assessment_tool/inex_eval/inex_eval/inex_eval.jar (2009)

Usage : *java -jar /smart/IR2010/2010_assessment_tool/2010-assessment-tool/inex_eval.jar -f qrels_file 1500_SubtoFOL.txt > MAIPvalue.txt*

Inputs : *-f – For Focused task*

qrels_file – Location of qrels file given by INEX

1500_SubtoFOL.txt – Chopped SubTOFOL file

Outputs *: MAIPvalue.txt - It prints the output on cmd which can be redirected to this file*

Example command for 2009 task :

java -jar /smart/IR2010/2009_assessment_tool/inex_eval/inex_eval/inex_eval.jar -f /smart/IR2010/2009_assessment_tool/inex_eval/inex_eval/inex2009.qrels s_1500_SubtoFOL.txt > MAIP.txt

Example command for 2010 task :

e.g., java -jar /smart/IR2010/2010_assessment_tool/2010-assessment-tool/inex_eval.jar -f /smart/IR2010/2010_assessment_tool/2010-assessment-tool/inex2010.qrels 1500_SubtoFOL.txt > MAIPvalue.txt

For Relevance In Context (RiC) task :

1. Removing mts from Flex output

The output of Flex contains some mts. To remove these mts, use the script, ***remove_mt_from_flex_output.pl***

Location *: /smart/10_scrpts/remove_mt_from_flex_output.pl*

Usage *: perl /smart/10_scrpts/remove_mt_from_flex_output.pl flex_output_file flex_output_with_no_mts*

Inputs *: flex_output_file – Output of Flex*

Outputs *: flex_output_with_no_mts – Output of this script*

e.g., perl /smart/10_scrpts/remove_mt_from_flex_output.pl flex_out.txt *no_mt_flex.txt*

2. <u>Removing overlaps</u> :

For focused evaluation, the overlaps are not allowed. The output of Flex is a ranked list of elements which might contain some overlaps. This step selects an element with higher correlation.

<u>Location</u> : /smart/10_scrpts/generate_focused_correlation_strategy.pl
<u>Usage</u> : /smart/10_scrpts/generate_focused_correlation_strategy.pl *no_mt_flex.txt* output.txt
<u>Inputs</u> : *no_mt_flex.txt – Flex-output with no mts (Output of step 1 in evaluation)*
<u>Outputs</u> : *output.txt – Output file generated after removing overlaps*

Example command for correlation strategy :
e.g., perl /smart/10_scrpts/generate_focused_correlation_strategy.pl no_mt_flex.txt correlation_strategy.txt
Example command for child strategy :
e.g., perl /smart/10_scrpts/generate_focused_child_strategy.pl no_mt_flex.txt child_strategy.txt
Example command for section strategy :
e.g., perl /smart/10_scrpts/generate_focused_section_strategy.pl no_mt_flex.txt section_strategy.txt

3. <u>Converting Flex output to INEX form</u>

The output of Flex is a ranked list of elements. To evaluate it using INEX, it should be converted to INEX format using the script : ***convert_to_inex_v2007.pl***.

<u>Location</u> : */smart/10_scrpts/convert_to_inex_v2007.pl*

Usage : perl /smart/10_scrpts/convert_to_inex_v2007.pl 72 focused Focused element 80000 2010001 section_strategy.txt flex_inex_format.xml

Inputs : 72 – Participant id

focused – Task id

Focused – Task id (e.g., UMD_FOCUSED)

element – Identifier for evaluation of all element retrieval

80000 – Max. number of elements for all queries

2010001 – First query id

no_mt_flex.txt – Flex-output without mts (output of previous step)

Outputs : flex_inex_format.xml – Flex output converted to INEX format

e.g., perl /smart/10_scrpts/convert_to_inex_v2007.pl 72 focused Focused element 80000 2010001 section_strategy.txt flex_inex_format.xml

4. Converting Flex output to TREC format

The output of above step generates output in xml format. The INEX eval tool needs the output in TREC format. It has xpaths for all elements and corresponding document and element ids. To convert this into TREC format use the script : ***XMLtoTRECformat.pl***

Location : /smart/10_scrpts/XMLtoTRECformat .pl

Usage : perl /smart/10_scrpts/XMLtoTRECformat .pl flex_inex_format.xml 1 80000 trec_format.txt

Inputs : flex_inex_format.xml – Flex output converted to INEX format (output of previous step)

1 – Task id (e.g., UMD_FOCUSED)

80000 - Max. number of elements for all queries

Outputs : trec_format.txt – Output TREC format *e.g., perl /smart/10_scrpts/XMLtoTRECformat.pl flex_inex_format.xml 1 80000 trec_format.txt*

5. Patching the xpaths

The previous step output has xpaths of all elements containing the tags which are not scrubbed off, e.g., article, bdy, sec, para etc. To convert these xpaths to ones with all intermediate tags use the script : ***ExpandXpathsNewMethod.pl***

Location *: /smart/10_scrpts/ExpandXpathsNewMethod.pl*

Usage *: perl /smart/10_scrpts/XMLtoTRECformat .pl trec_format.txt expandedFile.txt extended_doctrees_folder*

Inputs *: trec_format.txt – TREC format output (output of previous step)*

extended_doctrees_folder – location of extended doctrees for entire collection

Outputs *: expandedFile.txt – Output file having xpaths comtaining all tags*

e.g., perl /smart/10_scrpts/ExpandXpathsNewMethod.pl trec_format.txt expandedFile.txt /smart/chit0051/doctrees_extended1

All steps are common for 2009 and 2010 Focused task till here.

6. Convert xpaths into FOL formats :

Convert expanded xpaths to FOL formats using INEX SUBTOFOLeconomy jar.

Example command for 2009 task :

e.g., java -Xmx16G -jar /smart/IR2010/2009_assessment_tool/assessment-tool/2009-sub2fol-all/sub2fol/SUB2FOL.jar s_expandedFile.xml sub2FOL.txt

/smart/IR2010/2009_assessment_tool/assessment-tool/2009-sub2fol-all/sub2fol/gpxrai50rel.table.serialized

Location : */smart/IR2010/2009_assessment_tool/assessment-tool/2009-sub2fol-all/sub2fol/SUB2FOL.jar*

Usage : */smart/IR2010/2009_assessment_tool/assessment-tool/2009-sub2fol-all/sub2fol/SUB2FOL.jar expandedFile.txt SubtoFOL.txt location-of-gpxrai.table*

Inputs : *expandedFile.txt – File having xpaths with all tags (output of previous step)*

location-of-gpxrai.table – Location of gpxrai table given by INEX

Outputs : *SubtoFOL.txt – Output in FOL format*

Example command for 2010 task :

e.g., java -Xmx16G -jar /smart/IR2010/2010_assessment_tool/2010-assessment-tool/SUB2FOLeconomy.jar expandedFile.txt SubtoFOL.txt /smart/IR2010/gpxrai.table

Location : */smart/IR2010/2010_assessment_tool/2010-assessment-tool/SUB2FOLeconomy.jar*

Usage : *java -Xmx16G -jar location-of-SUB2FOLeconomy.jar expandedFile.txt SubtoFOL.trec++ location-of-gpxrai.table*

Inputs : *expandedFile.txt – File having xpaths with all tags (output of previous step)*

location-of-gpxrai.table – Location of gpxrai table given by INEX

Outputs : *SubtoFOL.trec++ – Output in FOL format*

7. Removing negatives from FOL (for 2009 and 2010 task) :

Now, in modified Flex we are not specifying the number of elements to be retrieved. Instead, Flex retrieves all relevant elements and we chop most relevant elements from it. There are few elements for which has correlation value as negative in FOL. We remove these elements and push the elements up to fill the window. The script used for this is : ***FOLRemoveNegatives.pl***

<u>Location</u> : /smart/10_scrpts/FOLRemoveNegatives.pl

<u>Usage</u> : perl /smart/10_scrpts/FOLRemoveNegatives.pl 1500 SubtoFOL.trec++ 1500_SubtoFOL.txt

<u>Inputs</u> : 1500 – Length of window, i.e., number of elements per query (For 2010 RIC tasks this number would be 8000)

SubtoFOL.txt – Output in FOL format

<u>Outputs</u> : 1500_SubtoFOL.txt – Output file with 1500 top elements per query

e.g., perl /smart/10_scrpts/FOLRemoveNegatives.pl 1500 SubtoFOL.trec++ 1500_SubtoFOL.txt

The steps are common for RiC 2009 and 2010 task and 2010 RRiC task till this point.

<u>For RiC 2009 task</u> :

8. <u>Use the INEX tool for evaluation (for 2009 task)</u> :

<u>Location</u> : /smart/IR2010/2009_assessment_tool/inex_eval/inex_eval/inex_eval.jar (2009)

<u>Usage</u> : java -jar /smart/IR2010/2009_assessment_tool/inex_eval/inex_eval/inex_eval.jar -r qrels_file 1500_SubtoFOL.txt > magpvalue.txt

<u>Inputs</u> : -r – For RiC task

qrels_file – Location of qrels file given by INEX

1500_SubtoFOL.txt – Chopped SubTOFOL file

<u>Outputs</u> : magpvalue.txt - It prints the output on cmd which can be redirected

to this file

Example command for 2009 task :

java -jar
/smart/IR2010/2009_assessment_tool/inex_eval/inex_eval/inex_eval.jar -f
/smart/IR2010/2009_assessment_tool/inex_eval/inex_eval/inex2009.qrels
s_1500_SubtoFOL.txt > MAIP.txt

For RiC 2010 task :

8. This step is done for 2010 tasks. This step removes elements from sub2fol if the number of irrelevant characters between the previous element and this element is >300 and it retrieves 1500 elements per query.

Script used : ***RIC_T2I300.pl***

Location : /smart/10_scrpts/ *RIC_T2I300.pl*
Usage : /smart/10_scrpts/ *RIC_T2I300.pl subtoFOL_80000.txt 300 1500 RIC_sub2Fol_CO_1500.trec++*
Inputs : *subtoFOL_80000.txt – The output of previous step having 80000 (maximum) elements per topic*
300 – Number of irrelevant characters between current and previous element
1500 – Number of elements to be retrieved per query
Outputs : *RIC_sub2Fol_CO_1500.trec++ – Output file generated having 1500 elements per topic*

e.g., perl /smart/10_scrpts/*RIC_T2I300.pl subtoFOL_80000.txt 300 1500 RIC_sub2Fol_CO_1500.trec++*

9. Use the INEX tool for evaluation (for 2010 task) :

Location : */smart/IR2010/2010_assessment_tool/2010-assessment-tool/inex_eval.jar (2010)*

Usage : *java -jar /smart/IR2010/2010_assessment_tool/2010-assessment-tool/inex_eval.jar -i qrels_file 1500_SubtoFOL.txt > magpvalue.txt*

Inputs : *-i – For RiC task*

qrels_file – Location of qrels file given by INEX

1500_SubtoFOL.txt – Chopped SubTOFOL file

Outputs : *magpvalue.txt - It prints the output on cmd which can be redirected to this file*

Example command for 2010 task :

e.g., java -jar /smart/IR2010/2010_assessment_tool/2010-assessment-tool/inex_eval.jar -i /smart/IR2010/2010_assessment_tool/2010-assessment-tool/inex2010.qrels 1500_SubtoFOL.txt > MAIPvalue.txt

For Restricted Relevance in Context (RRiC) task (2010 task) :

8. This step is done for 2010 RRIC task. This step removes elements from sub2fol if the number of irrelevant characters between the previous element and this element is >300. The T2I strategy is applied for all elements in a query.

Script used : ***RRIC_T2I300.pl***

Location : /smart/10_scrpts/ *RRIC_T2I300.pl*

Usage : /smart/10_scrpts/ *RRIC_T2I300.pl subtoFOL_80000.txt 300 80000*

RRIC_sub2Fol_CO_80000.trec++

Inputs *: subtoFOL_80000.txt – The output of previous step having 80000 (maximum) elements per topic*

300 – Number of irrelevant characters between current and previous element

80000 – Maximum number of elements to be retrieved per query

Outputs : *RRIC_sub2Fol_CO_80000.trec++ – Output file generated after applying T2I strategy for all the elements retrieved per query*

e.g., perl /smart/10_scrpts*/RRIC_T2I300.pl subtoFOL_80000.txt 300 80000 RIC_sub2Fol_CO_80000.trec++*

9. This step is done for 2010 tasks (RRIC only). This step restricts each element to exactly 500 characters and also retrieves 1500 characters per query

Script used : ***RRIC_exactly500.pl***

Location : /smart/10_scrpts/ *RRIC_exactly500.pl*

Usage : /smart/10_scrpts/ *RRIC_exactly500.pl RRIC_sub2Fol_CO_80000.trec++ 500 1500 RRIC_sub2Fol_CO_1500_exactly500.trec++*

Inputs : RRIC_sub2Fol_CO_80000.trec++ – The output of previous step (step 10)

500 – Number of characters per element

1500 – Number of characters to be retrieved per query

Outputs : RIC_sub2Fol_CO_1500.trec++ – Output file generated having 1500 characters per topic

e.g., perl /smart/10_scrpts/RRIC_exactly500.pl
RRIC_sub2Fol_CO_80000.trec++ 500 1500
RRIC_sub2Fol_CO_1500_exactly500.trec++

9. Use the INEX tool for evaluation (for 2010 task) :

Location : */smart/IR2010/2010_assessment_tool/2010-assessment-tool/inex_eval.jar (2010)*

Usage : *java -jar /smart/IR2010/2010_assessment_tool/2010-assessment-tool/inex_eval.jar -i qrels_file 1500_SubtoFOL.txt > magpvalue.txt*

Inputs : *-i – For RiC task*
qrels_file – Location of qrels file given by INEX
1500_SubtoFOL.txt – Chopped SubTOFOL file

Outputs : *magpvalue.txt - It prints the output on cmd which can be redirected to this file*

Example command for 2010 task :

e.g., java -jar /smart/IR2010/2010_assessment_tool/2010-assessment-tool/inex_eval.jar -i /smart/IR2010/2010_assessment_tool/2010-assessment-tool/inex2010.qrels 1500_SubtoFOL.txt > MAIPvalue.txt

References :

[1] Acquilla, N. Improving Results for INEX Focused Tasks, MS Thesis, Department of Computer Science, University of Minnesota Duluth, 2011.
http://www.d.umn.edu/cs/thesis/acquilla.pdf

[2] Arvola, P., Geva, S., Kamps, J., Schenkel,R., Trotman, A., and Vainio, J. Overview of the INEX 2010 Ad Hoc Track

http://www.cs.otago.ac.nz/homepages/andrew/2010-13.pdf

[3] Banhatti, R. Improving Results for the INEX 2009 Thorough and 2010 Efficiency Tasks, MS Thesis, Department of CS, UMD, August 2011
http://www.d.umn.edu/cs/thesis/banhatti.pdf

[4] Ganapathibhotla, M. Query Processing in a Flexible Retrieval Environment, MS Thesis, Department of CS, UMD, July 2006
http://www.d.umn.edu/cs/thesis/ganapathibhotla.pdf

[5] Geva, S., Kamps, J., Lethonen, M., Schenkel, R., Thom, J. and Trotman, A. Overview of the INEX 2009 Ad Hoc Track
http://www.cs.otago.ac.nz/homepages/andrew/2009-13.pdf

[6] INEX website, About INEX
https://inex.mmci.uni-saarland.de/about.html

[7] INEX website, Ad Hoc Retrieval Task and Result Submission Specification
http://www.inex.otago.ac.nz/tracks/adhoc/runsubmission.asp?action=specification

[8] INEX website, Document Collection
http://www.inex.otago.ac.nz/data/documentcollection.asp

[9] INEX website, INEX 2009 Guidelines for Topic Development
http://www.inex.otago.ac.nz/tracks/adhoc/gtd.asp

[10] Khanna S. Design and Implementation of a Flexible Retrieval System, MS Thesis, Department of Computer Science, University of Minnesota Duluth, 2005.
http://www.d.umn.edu/cs/thesis/khanna.pdf

[11] Narendravarapu, R. Improving Results for the INEX 2009 and 2010 Relevant in Context Tasks, MS Thesis, Department of CS, UMD, August 2011.
http://www.d.umn.edu/cs/thesis/narendravarapu.pdf

[12] Salton, G., Wong, A., Yang, C. A Vector Space Model for Automatic Indexing, *Comm. ACM*, 18(11), 613-620, 1975.

[13] Salton, G., editor. *The SMART Retrieval System – Experiments in Automatic Document Retrieval*, Prentice Hall, Englewood Cliffs, NJ, 1971.