Report on the Lucene4IR workshop: Developing Information Retrieval Evaluation Resources using Lucene (L4IR2016)

Leif Azzopardi[1], Yashar Mosfeghi[2], Martin Halvey[1],
Krisztian Balog[3], Emanuele Di Buccio [4],
Diego Ceccarelli[5], Juan Manual Fernandez Luna [6],
Charlie Hull[7], Jake Mannix[8], Sauparna Palchowdhury[9]

[1] University of Strathclyde {*Leif.Azzopardi,Martin.Halvey*}*@strath.ac.uk*
[2] University of Glasgow *Yashar.Mosfeghi@glasgow.ac.uk*
[3] University of Stavanger *krisztian.balog@uis.no*
[4] University of Padova *dibuccio@dei.unipd.it*
[5] Bloomberg *cdceccarelli4@bloomberg.net*
[6] University of Granda *jmfluna@decsai.ugr.es*
[7] Flax *charlie@flax.co.uk*
[8] LucidWorks *jake.mannix@lucidworks.com*
[9] NIST *sauparna.palchowdhury@nist.gov*

September 28, 2016

Abstract

The workshop and hackathon on developing Information Retrieval Evaluation Resources using Lucene (L4IR) was held on the 8th and 9th of September, 2016 at the University of Strathclyde in Glasgow, UK and funded by the ESF Elias Network. The event featured three main elements: (i) a series of keynote and invited talks on Lucene in action in industry, in teaching and learning environments, and evaluation forums. (ii) planning, coding and hacking where a number of groups created modules and infrastructure to use Lucene to undertake TREC based evaluations. And (iii) a number of breakout groups discussing challenges, opportunities and problems in bridging the divide between academia and industry, and how we can use Lucene and the resources created in teaching and learning IR evaluation. The event was composed of a mix and blend of academics, experts and students wanting to learn, share and create evaluation resources for the community. The hacking was intense and the discussions lively creating the basis of many useful tools and raising numerous issues. However, by adopting and contributing to most widely used and supported Open Source IR toolkit, it was clear that there were many benefits for academics, students, researchers, developers and practitioners - providing a basis for stronger evaluation practices, increased

reproducibility, more efficient knowledge transfer, greater collaboration between academia and industry, and shared teaching and training resources.

1 Introduction

Lucene and its expansions, Solr and ElasticSearch, represent the major open source Information Retrieval toolkits used in Industry. However, there is a lack of coherent and coordinated documentation that explains from an experimentalist's point of view how to use Lucene to undertake and perform Information Retrieval Research and Evaluation. In particularly, how to undertake and perform TREC based evaluations using Lucene. Consequently, the objective of this event was to bring together researchers and developers to create a set of evaluation resources showing how to use Lucene to perform typical IR operations (i.e. indexing, retrieval, evaluation, analysis, etc.) as well as how to extend, modify and work with Lucene to extract typical statistics, implement typical retrieval models. Over the course of the workshop participants shared their knowledge with each other creating a number of resources and guides along with a road map for future development.

2 Keynotes and Invited Talks

During the course of the workshops a series of talks on how Lucene is being used in Industry, Teaching and for Evaluation along with more technical talks on the inner workings of how Lucene's scoring algorithm works and how learning to rank is being included into Solr[1].

Introduction Talk: Why are we here?

Leif Azzopardi, University of Strathclyde: Leif explained how after attending the lively Reproducibility workshop [?] at ACM SIGIR 2015, he wondered where the Lucene team was, and why, if Lucene and the community is so big, why they don't come to IR conferences - he posited that perhaps we haven't been inclusive and welcoming as we could to such a large community of search practitioners. He further asserted that this has lead to a failure in transferring our knowledge into one of the largest open source toolkits available. He argued that if we as academics want to increase our impact then we need to improve how we transfer our knowledge to industry. One way is working with large search engines, but what about other industries and organisations that need search and use toolkits like Lucene? He argued that we need to start speaking the same language i.e. work with Lucene et al and look for opportunities on how we can contribute and develop resources for training and teaching IR and how to undertake evaluations and data science using widely used, supported and commonly accepted Open Source toolkits. He described how this workshop was a good starting point and opportunity to explore how academia and industry can better work together, where we can identify common goals, needs and resources that are needed to foster this relationship.

[1] Slides are available from `www.github.com/leifos/lucene4ir`

Keynote Talk: Apache Lucene in Industry

Charlie Hull, Flax: In his talk, Charlie first introduced Flax, and how it evolved over the years. Charlie explained that they have been building search applications using open search software since 2001. Their focus is on building, tuning and supporting fast, accurate and highly scalable search, analytics and Big Data applications. They are partners with Lucidworks, leading Lucene specialists and committers. When Lucene first came out clients were at reluctant to adopt open source, but nowadays it has been much more acceptable. Charlie notes that now you don't have to explain to clients what open source software is, and why it should be used. He described how Lucene-based search engines have risen in use - and that search and data analytics are available to those without six figure budgets. Charlie points out that Lucene is appealing because it is the most widely used open source search engine, which is hugely flexible, feature rich, scalable and performant. It is supported by a large and healthy community and backed by the Apache Software Foundation. Many of world's largest companies use Lucene including Sony, Siemens, Tesco, Cisco, Linkedin, Wikipedia, WordPress and Hortonworks. Charlie notes that they typically don't use Lucene directly, instead they use the search servers, built on top of Lucene, i.e. Apache Solr (which is mature, stable, and crucially highly scalable), or ElasticSearch (easy to get started with, great analytics, scalable). He contrasts these products with some of the existing toolkits in IR, and remarks on the latter, that "no one in industry has ever heard of them!". So even though they have the latest research encoded within them, it is not really viable for businesses to adopt them, especially as support for such toolkits is highly limited. He recommends that IR research needs to be within Lucene-based search services for it to be used and adopted.

Based on Charlie's experience he provided us with a number of home truths:

- Open source does not mean cheap
- Most Search engines are the same (in terms of underlying features and capabilities)
- Complex features are seldom used - and often confusing
- Search testing is rarely comprehensive
- Good search developers are hard to find

Charlie reflected on these points considering how we can do better. First, learn what works in industry and how industry are using search - there are lots of research challenges which they rarely get to solve and address but solutions to such problems would have real practical value. Second, improve Lucene et al with ideas from academia - faster - for example, it took years before BM25 replaced TFIDF as the standard ranking algorithm, where as toolkits like Terrier already have infrastructure for Learning to Rank, while this is only just being developed in Lucene. Third, he pointed out that testing and evaluation of Lucene based search engines is very limited, and that thorough evaluations by search developers is poor. He argued that this could be greatly improved, if academics and researchers, contributed to the development of evaluation infrastructure, and transferred their knowledge to practitioners on how to evaluate. Lastly, he pointed that the lack of skilled and knowledgable search developers was problematic - having experience with Lucene, Solr and ElasticSearch are highly marketable skills, especially, when there is a growing need to process larger and larger volumes of data - big data requires data scientists! So there is the pressing need to create educational resources and training material for both students and developers.

Using Lucene for Teaching and Learning IR: The University of Granada case of study

Prof. Juan M. Fernndez-Luna (University of Granada) : In his talk, Juanma explained the Bologna Process, the establishment of the European Higher Education Area, and how the University of Granada (UGR) has adopted its study programmes, changing the existing undergraduate and master degrees and introducing a few new ones. Currently, Computer Science studies at UGR are composed of an undergraduate degree of four years and a master degree of one, with three different options in this last case: one professional master in Computer Science and two research masters (Data Science and Computer Engineering, and Software Development). With a lot more attention focused on Information Retrieval, a number of courses have been introduced within their undergraduate and masters courses:

- *Information Retrieval* at the undergraduate Computer Science degree (6 ECTS, 4th year). The objective of this subject is that the students learn the foundation of IR (document preprocessing, indexing, retrieval models, evaluation and text classification and clustering).
- *Information Management in the Web* at Master in Computer Science (4 ECTS). This subject is composed of three parts: social network analysis, IR and recommender systems. The basic aim is to show the students different ways of managing and accessing the information in the Web. The part related to IR is focused just in briefly explaining the IR foundations.
- *Information Retrieval and Recommender Systems* at Master in Data Science and Computer Engineering (3 ECTS). In the context of this master, IR foundations are shown to the students, but with a perspective closer to Data Science (preprocessing of large document collections, clustering and classification, etc.).
- *Undergraduate and Master Thesis* (12 ECTS). Each degree contains a final thesis where the students have to show the skills they have acquired during their studies by means of the development of a project. These are proposed by the lecturers and some of them are related to Information Retrieval.

Juanma explained that before introducing programming details, the lecturers at UGR thought that it would be better if students could understand the core IR process itself. To facilitate the teaching and learning process, *SulaIR* [**?**] was designed. This is a desktop tool that covers the different IR stages: web crawling, document pre-processing, indexing, retrieval and relevance feedback. The tool lets students interact with all these processes and secure the concepts from a practical point of view.

In any of these IR-related subjects, the same question came about when the lecturers were planning lab work and exercises, is it better to: (1) create IR projects from the scratch, programming even the more basic classes and focusing on the implementation details, or (2) use an existing IR library (Lucene, Terrier, Lemur, MG, etc.) and focusing on the process?

At the very beginning, they opted for the first alternative as they thought that this could help to understand the details of the search engines, but this created to two learning challenges. The students had to understand the IR process, in a first abstraction step, and then, to transfer it to code, in a second step. Often the second step interfered with their understanding of the first step as they faced many programming challenges. So the learning process was not very effective.

Therefore, Juanma and the lecturers made the decision of using an Open Source library, where the problem is reduced to learning about an API and identifying the classes and methods that need to be used. In this case, students need not care about the implementation details, per say, instead they could focus on the process (or at least this was their theory). In addition, they realised that the typical professional developer with real needs regarding IR will use an API/Toolkit and there will not be programming IR-related modules from scratch. From all of available APIs, Lucene was, without any doubt, the first choice. This was because it is the most popular IR toolkit openly available and most widely used in industry.

Juanma points out that Teaching and Learning IR with Lucene in these subjects is not without its own problems and challenges. After using it with their courses, the lecturers came to a number of realisations:

- From *Information Retrieval*:
  - Lucene is a "monster", with lots of classes and methods. This is because it is a large-scale production system, and so students often are frighten by it, unsure of how to work with it.
  - Considering Java as the programming language to work with Lucene, our students usually have to learn this language first, as they are used to work with C++ in the degree. This language could be a possibility but the Lucene C++ API is poorly documented in comparison to its Java version, so it was discarded.
  - There is a large amount of documentation about how basic tasks are carried out with Lucene, but sometimes the students select sources from different versions of Lucene. And this causes many headaches when understanding and debugging the problems faced when programming with Lucene.
  - The students' learning curve is very steep when working with the basic process. However, when they progress to more advanced topics such as fielded and complex queries, they find that they can advance at a good pace.
- From *Information Management in the Web*:
  - Due to the length of the course, the ABCs of IR with Lucene are shown to the students. They consider this a good approach because in case of needing to build a search engine in their professional career, they have the basic knowledge to use a toolkit to configure a search engine.
- From *Information Retrieval and Recommender Systems*:
  - In the context of a Data Science Master, students are not so interested in the retrieval process itself, but in the pre-processing and indexing stages as bases of further tasks related to machine learning. Other toolkits, as Tika, are shown.
  - Students have different programming backgrounds (Computer Science, Statistics, Mathematics, Information Science, Electronics, etc.), so it is a real problem to use Lucene for developing lab projects. Considering that R is the programming language on which most of the subjects from the Data Science master are based, an initiative such as RLucene[2] could be very interesting for the students.
- From *Undergraduate and Master Thesis*:

[2] https://github.com/s-u/RLucene

- It is difficult to find Lucene advanced documentation for more specific topics, so they usually spend too much time trying to work out how such functionality works (usually through trial and error). In addition, it is very difficult for them to develop new retrieval models or ranking functions.
- As Lucene has not got native support for documents, they have to make great efforts to build the piece of software required to extract the text of the real documents that they find in their projects.
- There is no support for IR evaluation in Lucene, so students have to write their own tools to evaluate the performance.

As a conclusion, Juanma and the other lecturers at UGR recognised that Lucene is a great tool for teaching and learning IR, but clearly the experience would be much more productive if:

- Lucene documentation or tutorials from the point of view of teaching and learning were available as well as material describing advanced tasks.
- Visual and/or Command-Line Tools for teaching and learning the IR process based on Lucene, a kind of SulaIR-L, would really be very useful.
- Visual and/or Command-Line Tools for IR evaluation (TREC-based) were available, or at least, classes for these purposes were included in the API.

Black Boxes are Harmful

Sauparna Palchowdhury (NIST):Having seen students and practitioners in the IR community grapple with abstruse documentation accompanying search systems and their use as a black box, Sauparna, in his talk, argued why Lucene is a useful alternative and how and why we must ensure it does not become another black box. In establishing his views, he described the pitfalls in an IR experiment and the ways of mitigation. The suggestions he puts forth, as a set of best practices, highlights the importance of evaluation in IR to render an experiment reproducible and repeatable and the need for a well-documented system with correct implementations of search algorithms traceable to a source in IR literature. In the absence of such constraints on experimentation students are misled and learn little from the results of their experiments and it becomes hard to reproduce the experiments. As an example, the talk cited a wrong implementation of the *Okapi BM25* term-weighting equation in a popular research retrieval system (Table 2). Following this was a brief how-to on implementing *BM25* (or any TFxIDF weighting scheme) in Lucene (Table 2). This also explained Lucene's way of computing the similarity between two text documents (usually referred to as 'Lucene's scoring') that may be of use to the student and practitioner interested in using Lucene for IR experiments.

Some of the points of failures mentioned in the talk are misplaced test-collection pieces (document-query-qrel triplet), counterintuitive configuration interfaces of systems, poor documentation that makes systems look enigmatic and lead to the creation of heuristics passed around by word-of-mouth, naming confusion (a myriad of TFxIDF model names), blatant bugs and not knowing how the parser works. As mitigation, Sauparna listed some of the things he did as an experimenter. He wrote a script (TRECBOX) to abstract the IR experiment pipeline and map them to configuration end-points of the three systems; Terrier, Lucene and Indri. This would enable documenting an experiment's design in plain text files, that could be shared and the experiment repeated. He constructed a survey of TFxIDF variants titled *TFxIDF Repository* [?] meant to be a single point of reference to help disambiguate the variants in the wild. All mentions of term-weighting equations in this repository are traceable to a source in IR literature. He also shows how to visually juxtapose evaluation results obtained using a permutation of a set of systems, retrieval models and test-collections on a chart that would act as a sanity check for the system's integrity. As a part of these investigations he modified Lucene for use with TREC collections (the mod was named LTR) which is available for others to use. The "mod" is also accompanied by notes to augment Lucene's documentation. The gamut of Sauparna's work is to be found online [?].

Lucene's documentation does not use well-defined notation to represent its way of computing the similarity score between a pair of documents. The notation below is Lucene's scoring equation, lifeted from Lucene's documentation. It uses actual function names that are to be found in Lucene's source code;

$$score(Q, D) = coord(Q, D) \cdot qnorm(Q) \cdot \sum_{T \in Q} (tf(T \in D) \cdot idf(T)^2 \cdot boost() \cdot norm(T, D))$$

Sauparna's explanation begins with a well-defined, generalized, notation for Lucene's scoring in step with the definition from Lucene's documentation [?];

$$score(Q, D) = f_c(Q, D) \cdot f_q(Q) \cdot \sum_{T \in Q \cap D} (tf(T_k) \cdot df(T_k) \cdot f_b(T_k) \cdot f_n(T_k, D)))$$

He picks two popular TFxIDF variants, breaks them down into meaningful components (a term-frequency transformation, a transformation on the document-frequency and a length normalization coefficient) and plugs these components into Lucene's equation. The components in Lucene's equation that are left unused are replaced by the integer 1, meaning, the functions return 1; which has no effect on the chain of multiplications. Table 2 lists the variants and components and Table 2 shows where the components were transplanted to.

Making a reference to the SIGIR 2012 tutorial on *Experimental Methods for Information Retrieval* [?], Sauparna states that we need to take a more rigorous approach to the IR experimental methodology. A list of best practices was recommended that would add more structure to IR experiments and prevent the use of systems as black boxes. These were:

- Record test-collection statistics.
- Provide design documentation for systems.
- Use a consistent naming scheme and a well-defined notation.
- Build an evaluation table to be used as a sanity-check.
- Isolate sharable experimental artefacts.
- Ensure that implementations are traceable to a source in IR literature.

In conclusion, Sauparna suggests that if we, the IR research community, are to build and work with Lucene, then it would be helpful to consider these points when introducing new features into Lucene.

Deep Dive into the Lucene Query/Weight/Scorer Java Classes

Jake Mannix, Lucidworks: In this more technical talk, Jake explained how Lucene scores a query, and what classes are instantiated to support the scoring. Jake described, first, at a high level how to do scoring modification to Lucene-based systems, including some

TFxIDF Variants: what's correct and what's not.

Name	w_{ik}	w_{jk}
BM25(A)	$\frac{f_{ik}}{k_1((1-b)+b\frac{dl_i}{avdl})+f_{ik}} \times \log(\frac{N-n_k+0.5}{n_k+0.5})$	$\frac{(k_3+1)f_{jk}}{k_3+f_{jk}}$
BM25(B)	$\frac{(k_1+1)f_{ik}}{k_1((1-b)+b\frac{dl_i}{avdl})+2f_{ik}} \times \log(\frac{N-n_k+0.5}{n_k+0.5})$	$\frac{(k_3+1)f_{jk}}{k_3+f_{jk}}$
Okapi BM25	$\frac{(k_1+1)f_{ik}}{k_1((1-b)+b\frac{dl_i}{avdl})+f_{ik}} \times \log(\frac{N-n_k+0.5}{n_k+0.5})$	$\frac{(k_3+1)f_{jk}}{k_3+f_{jk}}$
components	$T \times I$	Q
SMART dtb.nnn	$\frac{(1+\log(1+\log(f_{ik}))) \times \log(\frac{N+1}{n_k})}{1-s+s \cdot \frac{b_i}{avgb}}$	f_{jk}
components	$T \times I \div L$	Q

Table 1: The similarity score; $score(D_i, D_j) = \sum_{k=1}^{t}(w_{ik} \cdot w_{jk})$ $\forall i \neq j$, combines the weight of a term k over the t terms which occur in document D_i and D_j. Since a query can also be thought of as a document in the same vector space, the symbol D_j has been used to denote a query without introducing another symbol, say, Q. BM25(A) and BM25(B) are the two incorrect implementations found in a popular retrieval system. Comparing them to *Okapi BM25* on the third row shows that A has the $k_1 + 1$ factor missing in the numerator, and B uses twice the term-frequency $2f_{ik}$ in the denominator. Neither can they be traced to any source in IR literature, nor does the system's documentation say anything about them. The *Okapi BM25* and the *SMART dtb.nnn* variants are known to be effective formulations developed by trial and error over eight years of experimentation at TREC 1 through 8. Their forms have been abstracted using capital letters to show how these components fit in Lucene's term-weight expression.

IMPLEMENTING TFxIDF VARIANTS IN LUCENE

Lucene	$f_c(Q,D)$	$\cdot$	$f_q(Q)$	$\cdot$	$\sum_{T \in Q \cap D}$	(	$tf(T_k)$	$\cdot$	$df(T_k)$	$\cdot$	$f_b(T_k)$	$\cdot$	$f_n(T_k, D_j)$	)
BM25	1	$\cdot$	1	$\cdot$	$\sum_{T \in Q \cap D}$	(	T	$\cdot$	I	$\cdot$	Q	$\cdot$	1	)
dtb.nnn	1	$\cdot$	1	$\cdot$	$\sum_{T \in Q \cap D}$	(	T	$\cdot$	I	$\cdot$	Q	$\cdot$	L	)

Table 2: Plugging components of the TFxIDF equation into Lucene's scoring equation; the first row is the generalized form and the following two rows show the components of two popular TFxIDF equations transplanted to Lucene's equation. Table 2 specifies what the capital letters represent.

"Google"-like questions on how to score efficiently. Then, he went into more details about the BooleanQuery class and is cousins, showing where the Lucene API allows for modifications of scoring with pluggable Similarity metrics and even deep inner-loop, where ML-trained ranking models could be instantiated - *if you're willing to do a little work.*

Learning to Rank with Solr

Diego Ceccarelli, Bloomberg On day two of the workshop, Diego started his talk by explaining that tuning Lucene/Solr et al is often performed by "experts" who hand tune and craft the weightings used for the different retrieval features. However, this approach is manual, expensive to maintain, and based on intuitive, rather than data. His working goal behind this project was to automate the process. He described how this motivated the use of Learning To Rank, a technique that enables the automatic tuning of a information retrieval system by applying machine learning when estimating parameters. He points out that sophisticated models can make more nuanced ranking decisions than a traditional ranking function when tuned in such a manner. During his talk, Diego presented the key concepts of Learning to Rank, how to evaluate the quality of the search in a production service, and then how the Solr plugin works. At Bloomberg, they have integrated a learning to rank component directly into Solr (and released the code as Open Source), enabling others to easily build their own Learning To Rank systems and access the rich matching features readily available in Solr.

3 Discussion

During the course of the workshop, two breakout groups were formed to discuss how we can use Lucene when teaching and learning, and what were the main challenges in bridging the industry/academia along with what opportunities it could bring about. Finally, we asked participants to provide some feedback on the event.

3.1 Teaching and Learning

To seed the discussion for this working group, various members explained how they ran their IR courses. As Juanma had already discussed how they teaching at the University of Granada (see above), others described their courses and experiences.

Emanuele Di Buccio, University of Padova: Emanuele described one of the courses taught as part of the Master Degree in Statistical Science at the University of Padua, called *Information Systems (Advanced)*[3]. The course covered both basic IR topics: indexing and retrieval methods, retrieval models, and evaluation, along with more advanced topics such as Web Search or Machine Learning for IR. A detailed description of the course contents can be found in [**?**], which is an IR book developed from the experiences teaching the course. The course was designed so that, for most of the topics, lessons at a theoretical level on a specific topic were followed by a laboratory assignment on that topic. The topics covered in laboratory assignments were:

- creation of a test collection,
- indexing,
- retrieval,
- relevance feedback,
- link analysis,
- Learning to Rank, and
- Optimization of Ranking functions with parameters.

Emanuele explained that students were asked to propose their own methodology to carry out the laboratory activities. For instance, when considering the topic of "relevance feedback", each student could propose their own methodology to perform feedback, e.g. through a query expansion method or term re-weighting. Each assignment, then, involved the experimental evaluation on a shared test collection. Indeed, the objective of the assignments was three-fold:

1. to better understand the topic;
2. to become familiar in the design and the implementation of experimental methodologies to evaluate methods and/or components and,
3. more generally, to test research hypotheses.

Students were allowed to use a manual approach (when possible), a software library or build their own software modules to achieve the assignment objective; a list of software libraries were provided before the first laboratory assignment to make the students aware of possible options. However, the adoption of a manual approach for some of the laboratory activities was mandatory. For instance, in the case of the assignment on indexing, the use of a manual approach aimed at a better understanding of the conceptual mechanisms to identify the most effective descriptors to retrieve relevant documents. When the students proposed their own methodology for indexing, they were asked to present their approach as a set of steps that can be automated.

The availability of software libraries or resources to easily use the basic operations is crucial to allow the students to test their methodology with little/less effort. In an edition of the course, a lesson was dedicated to a general introduction to Apache Lucene, where sample code was provided. Along with Apache Lucene, and introduction to ElasticSearch [**?**] was also presented, particularly how to index documents, perform retrieval, and how to customize

[3] The description refers to the course editions in the Academic Years 2011/12-2014/15. The professor in charge was Massimo Melucci.

Apps	High Level	Low Level
IndexerApp	Modify how the indexer is performed i.e. different tokenizers, parsers, etc	Can modify parsers, tokenizers, etc
IndexAnalyzerApp	Inspect the influence of indexer	
RetrievalApp	Try out different retrieval algorithms Change retrieval parameters	Implement new retrieval algorithms
trec_eval	Measure the performance	
ResultAnalyzerApp	Inspect and analyze the results returned	Customise the analysis, put out other statistics of interest
ExampleApp		Examples of how to work with the Lucene index, to make modifications
Batch Retrieval Scripts	Configure to run a series of standard batch experiments	Customize to run specific retrieval experiments
RetrievalShellApp	n/a	Customize to implement retrieval algorithms outwith the Lucene scorer i.e. a simple scorer assuming term independence

the scoring mechanism via scripting[4] The main reason for the introduction to ElasticSearch was that the students could index, retrieve, and customize the retrieval algorithm – and therefore test some of their methodologies – without writing actual code but only through the use of REST requests.

Another aspect Emanuele commented on was the heterogeneous background of the students, and how they came from various disciplines. This was one of the reasons, why they did not restrict the laboratory activities to a single software library and allowed students to select the tool they felt most comfortable with. While students in Computer Science and Computer Engineering were familiar with Java, students in Statistical Science tended to prefer the R language because it was used in many courses within their course degree. Therefore, one of the resources that could be useful for teaching is a R wrapper for Apache Lucene — wrappers in other programming languages exist, e.g. PyLucene [**?**] for Python. Software libraries such as ElasticSearch, could be useful tools to support teaching: for instance, they provide functionalities – in the event of ElasticSearch a REST request – to display how a specific fragment of text is processed given a pipeline, e.g. a specific tokenizer and a set of filters (lowercase, porter stemming, ...). **Krisztian Balog, University of Stavanger**:

explain overview of course, novel assessment methods, tools used, problems faced, etc

Martin Halvey, University of Strathclyde:

How do you go about teaching IR? What level?

What kinds of things do you need/want from such resources?

How do we see Lucene fitting in? Benefits to students?

3.2 Challenges and Opportunities

Some challenges from charlie - lack of good real-world test data for researchers. Companies need to provide this - many open source search companies are small and therefore find it hard to support apprenticeships, internships etc. or access funding such as KTPs - Lucene

[4]ElasticSearch allows to evaluate a custom score via scripts — see the *Scripting* module. Apache Solr provides similar functionalities via *Function Queries*.

community can be hard to enter - learning curve can be steep, this is a very large and complex project

3.3 Feedback

4 Resources

As part of the workshop numerous attendees contributed to the Lucene4IR GitHub Repository - `http://github.com/leifos/lucene4ir/`. In the repository, three main applications were developed and worked on:

- IndexerApp - enables the indexing of several different TREC collections, e.g. TREC123 News Collections, Aquaint Collection, etc.
- RetrievalApp - a batch retrieval application when numerous retrieval algorithms can be configured, e.g. BM25, PL2, etc
- ExampleStatsApp - an application that shows how you can access various statistics about terms, documents and the collection. e.g. how to access the term posting list, how to access term positions in a document, etc.

The repository also contained a sample test collection (documents, queries and relevance judgements) was provided (CACM), so that participants could try out the different applications.

During the workshop, a number of different teams undertook various projects:

- **Customisation of the tokenisation, stemming and stopping during the indexing process**: this enabled the IndexerApp to be configured so that the collections can be indexed in different ways i.e. they could change the stemming algorithm, include a stop list, enable positions to be recorded, etc. The idea being that students would be able to vary the indexing parameters and then see the effect on the collection and performance .
- **Implementation of other retrieval models**: inheriting from Lucene's BM25Similiarity Class, BM25 for Long documents was implemented BM25L [**?**], OKAPI BM25's was also implemented to facilitate the comparison between how it is currently implemented in Lucene versus an implementation of the original BM25 weighting function [**?**].
- **Alternative Scoring Mechanism**: Rather than scoring through Lucene's mechanics (Query $\rightarrow$ Weight $\rightarrow$ Scorer), others attempted to implement BM25 by directly accessing the inverted index through the Lucene's index API. The objective was twofold. First, to provide a "template" to implement a retrieval model where document matching is performed through a Document At A Time (DAAT) strategy and access to term vocabulary (via Terms and TermsEnum) and to posting lists (via PostingsEnum) is made more explicit; in some scenarios (e.g. for teaching activities) it could be useful to provide sample code that relies only on general concepts (term vocabulary, posting lists, etc.) and it is not tailored to specifics of the library — e.g. the Lucene scoring model. The second objective was to provide an "easier" way to control the variables in the experimental settings or to investigate if choices made for efficiency purposes (e.g. document length approximation) significantly affect effectiveness.

- **Query Expansion**: QueryExpansionRetrievalApp
- **Hacking the Inner-Loop**: The break-out group focused on inner loop scoring wanted to try something that was simultaneously simple, practical, and yet required some inner loop scoring magic. Based on the interests of the group members, we decided on "cross-field phrase queries": an extension of the idea of a sloppy phrase query where the "slop" allowed for a pair of terms occurring in *different* fields to be part of a phrase (but with a parametrizably lower score than terms in the same field). We worked out the design (delegating most of the work to Query / Weight / Scorer classes already in Lucene, but then combining them together across fields), and stepped through much of the iteration implementation. While we got most of the plumbing done, we only had enough time for our "score()" method to be implemented as naively as imaginable, and did not get it fully working in the time of the workshop. Some participants expressed interest in working on it further, to see how efficient it was, and what effect on scoring it would have (if a QueryParser was configured to explicitly spit out queries of this form sometimes).
- **Working with Lucene's Index and Reader**: Additional Examples on how to access and work with Lucene's index were also added to the ExampleStatsApp. These code snippets showed how it was possible to iterate through the term postings list, how to iterated through documents, and access various document, term and collection statistics. The purpose of the app was to demonstrate how to work with the Lucene index in order to perform various operations, which are often difficult to work out from the code base or existing documentation.

Along with the code some documentation was also produced to explain various aspects and how to set up and run the different applications. However, now, post workshop, there is the need to bring together all the elements developed and collate all the documentation so that these resources can be used for teaching and learning IR evaluation using Lucene by the community.

5 Summary

6 Acknowledgments

We thank the European Science Foundation / ELIAS Network for funding the workshop (Grant No. SM 5916). We would also like to thank our speakers as well as Bloomberg, FlaxSearch, LucidWorks and the University of Strathclyde. Finally, we would like to thank all the participants for their contributions to the workshops and hackathon. Also, thanks to Manisha, Guido and Casper for their offline contributions.

Add references to indri, lemur, terrier, lucene, etc, BM25L

References

[1] Apache Lucene. Class TFIDFSimilarity, 2016. [Online; accessed 21-September-2016].

[2] Arguello, J., Crane, M., Diaz, F., Lin, J., and Trotman, A. Report on the sigir 2015 workshop on reproducibility, inexplicability, and generalizability of results (rigor). *SIGIR Forum 49*, 2 (Jan. 2016), 107–116.

[3] Elastic. `https://www.elastic.co/products/elasticsearch`.

[4] Lv, Y., and Zhai, C. When documents are very long, bm25 fails! In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2011), SIGIR '11, ACM, pp. 1103–1104.

[5] Melucci, M. *Information retrieval. Metodi e modelli per i motori di ricerca.* Informatica: Nuova serie. Franco Angeli, 2013.

[6] Metzler, D., and Kurland, O. Experimental methods for information retrieval. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2012), SIGIR '12, ACM, pp. 1185–1186.

[7] Palchowdhury, S. IR, 2016. [Online; accessed 21-September-2016].

[8] Palchowdhury, S. TFxIDF Repository, 2016. [Online; accessed 21-September-2016].

[9] PyLucene. `http://lucene.apache.org/pylucene/`.