Simulating Interactive Information Retrieval

SimIIR: A Framework for the Simulation of Interaction

David Maxwell and Leif Azzopardi
School of Computing Science
University of Glasgow
Glasgow, Scotland
d.maxwell.1@research.gla.ac.uk
Leif.Azzopardi@glasgow.ac.uk

ABSTRACT

Simulation provides a powerful and cost-effective approach to explore and evaluate how interactions between a searcher and system influence search behaviour and performance. With a growing interest in simulation and an increasing number of papers using such an approach, there is a need for a flexible framework for simulation. Thus, we present *SimIIR*, an open-source toolkit for building and conducting *Interactive Information Retrieval (IIR)* experiments. The framework consists of a number of high level components, including the simulation, the searcher and the system, all of which must be configured. The SimIIR framework provides a series of interchangeable components. Examples of these components include the querying strategies (how simulated queries are formulated) and stopping strategies (the depth to which a searcher will examine snippets and documents) that a simulated searcher will employ. We have implemented various existing strategies so that they can be used by other researchers to not only replicate and reproduce past experiments, but also create new experiments. This paper describes the SimIIR framework and the different components that can be configured and extended as required.

Keywords: Simulation, Search Behavior, Strategy, Stopping Strategies, Continuation Strategies, Querying Strategies, User Modeling

1. INTRODUCTION

The process of search is inherently interactive. During a search session, a searcher can issue a number of queries, examine a number of snippets, and assess documents for relevance to their information need. Despite this, the *Information Retrieval (IR)* community has centred much of its research around the so-called *Cranfield Paradigm*. Revolving around the concept of standardised test collections and relevance judgements, the paradigm makes a number of key assumptions that are at odds with the interactive nature of search. Assumptions include a searcher: *(i)* issuing a single query; *(ii)* examining each document in turn and independently; and *(iii)* examining snippets and documents to a fixed depth. Simulation, as outlined by Keskustalo et al. [10], provides a means to go beyond the limitations presented by the Cranfield paradigm without conducting an expensive and time consuming user study. Simulation also offers a number of other benefits over user studies. The approach provides a rapid means to exploring *'what-if'* scenarios, and also facilitates a range of evaluations, such as a comparison between systems and understanding searcher behaviours [4]. Simulations enable a range and variety of searchers to be created who do not suffer from issues such as fatigue or learning effects, unless specifically coded to do so. Simulated searchers are therefore highly controlled, and can therefore yield reproducible results [4].

With a growing interest in simulation and a growing number of researchers employing the technique in their work (e.g. [1, 2, 3, 6, 7, 9, 13, 14, 15, 17]), we argue that there is a need for an IIR-based simulation toolkit. A toolkit freely available to the IIR community will ensure that experiments conducted with it can be easily reproduced, as well as reducing the major overheads for creating simulations. To address this issue, we introduce an open-source framework for the simulation of IIR (*SimIIR*). Over the past two years, we have been building the SimIIR framework and used it to produce a number of different simulations [13, 14] - with more to come. The framework can be configured in a variety of different ways to support different search processes, different searcher configurations, different experimental conditions, and different search engines.

2. SIMIIR

Previous studies employing simulation have either: *(i)* considered aspects of the IIR process in isolation, such as query generation (e.g. [11]); or *(ii)* considered the search process as a whole (e.g. [14]). SimIIR is designed to capture and broadly reflect the complex processes involved within the wider IIR process, and has been developed from the ground up in the *Python*[1] programming language using a highly modularised, class-based framework. Each component broadly represents a major stage in the IIR process[2],

[1] Python is a general purpose, high level programming language available at http://www.python.org. The SimIIR framework is presently designed to run on Python `2.7.x` environments.

[2] The SimIIR framework, complete with a variety of example simulation configuration files, is freely available at http://git.io/vZ5mH.

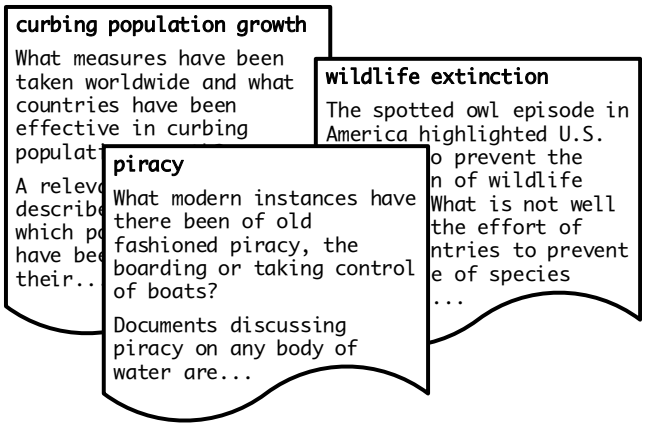

Figure 1: An example of three topic descriptions used within the SimIIR framework. The first line denotes the topic title, with everything following the title considered the topic description.

with new components simply inheriting from the relevant base class. This allows for the easy experimentation of different simulated searchers using a variety of test collections, querying strategies, stopping strategies, and more. This also highlights one of the main advantages of SimIIR in that it can be easily adapted to suit ever more advanced components as the research in this area progresses. SimIIR has already been employed in several publications investigating stopping strategies by Maxwell et al. [13, 14].

SimIIR Components: In order to successfully run an experiment on SimIIR, a *simulation* must first be defined. A simulation is comprised of a series of **topics**, a **search interface/engine**, an **output controller**, and a series of **simulated searchers** - all of which are elaborated on in subsequent sections. A simulation in essence is loosely associated with the concept of a real-world experiment, such as a user study. The experiment would be comprised of a series of searchers or subjects, examining documents for relevance over one or more topics. The parameters for each of these settings are specified in an XML *simulation configuration* file and passed to SimIIR. The experiment itself consists of a series of runs, the total number of which can be calculated by summing the combinations of the specified number of searchers and topics. The final component comprising a simulation is the list of simulated searchers. Every simulated searcher is specified within a separate XML *user configuration file*. Within this file, a host of additional components are defined, namely: a **querying strategy/generator**; a **classifier/decision maker** for both **snippets** and **documents**; a **stopping strategy**; a **logger**; and a **search context**. The components which comprise the searcher are encoded within the underlying **searcher model**, providing the various actions that are observed.

2.1 Topics

We define a topic as a description of what a simulated searcher is expected to find relevant. These are specified by a series of *topic description files*, consisting of a *title* and *description*. Figure 1 illustrates examples of such files. All the topic files that have been generated for SimIIR thus far are based upon the topics provided by various *TREC* tracks, such as the TREC *2005 Robust Track*.

```
ACTION  QUERY     1200.0  13.9  extinction wildlife
ACTION  SERP      1200.0  15.4  EXAMINE_SERP
ACTION  SNIPPET   1200.0  16.7  SNIPPET_REL       APW1998…
ACTION  DOC       1200.0  40.5  EXAMINING_DOC     APW1998…
ACTION  MARK      1200.0  43.1  CONSIDERED_REL    APW1998…
ACTION  SNIPPET   1200.0  44.4  SNIPPET_REL       XIE1998…
ACTION  DOC       1200.0  68.2  EXAMINING_DOC     XIE1998…
ACTION  SNIPPET   1200.0  69.5  SNIPPET_NR        NYT1999…
ACTION  QUERY     1200.0  83.4  efforts extinction wildlife
ACTION  SERP      1200.0  84.9  EXAMINE_SERP
ACTION  SNIPPET   1200.0  86.2  SEEN_PREV         XIE1998…
```

Figure 2: An example snippet of the output log file generated by SimIIR. Included is the action performed by the simulated searcher (e.g. `QUERY`, `SNIPPET`), the total time available (`1200` seconds), the elapsed time of the simulated session, and the judgement for the action (if applicable), such as `SNIPPET_REL` for when a snippet is considered relevant.

2.2 Search Interface/Engine

We consider the search interface/engine component as an abstraction of a search engine and the *Search Engine Results Page (SERP)*. After issuing the search interface/engine with a query, the component provides SimIIR with access to the SERP - a ranked list of snippets and associated documents. As the interface is highly genericised, any search engine with a Python wrapper can be easily coupled to the SimIIR framework. Presently, only a wrapper for the *Whoosh* IR toolkit[3] has been implemented. Any additional configuration options for a particular search interface/engine can also easily be set in the simulation configuration file.

2.3 Output Controller

SimIIR also provides a flexible output controller with a variety of output options for the simulations that are run with the framework. Options are based upon the output files that are saved for each simulated run, and include options to save the interaction log of the simulated searchers (e.g. queries issued, snippets examined, documents examined), and whether to save the list of documents considered relevant by the simulated searchers. The file specified by the final option may then in turn be fed into an evaluation program such as `trec_eval`[4] to obtain the values for the various measures and metrics that can be computed.

For each run, output files include: a `.log` file, which contains the complete set of interactions undertaken by the simulated searcher (refer to Figure 2 for an example of an output log); a `.cfg` file, containing the configuration of components for a given simulated searcher; an `.out` file, containing output from an evaluation program such as `trec_eval` (if enabled); a `.queries` file, containing a line break separated list of queries issued; and a `.res` file, containing a list of the documents considered relevant (in the format of a standard TREC results file). Runs are identified by unique identifiers specified in the simulation configuration and searcher configuration files to avoid results being overwritten.

2.4 Querying Strategies/Generators

The querying strategy/generator is the component responsible for the generation and selection of queries. A variety

[3] *Whoosh* is a pure Python indexing and search library, available at https://pypi.python.org/pypi/Whoosh.

[4] The `trec_eval` program is freely available to download and compile at http://trec.nist.gov/trec_eval/.

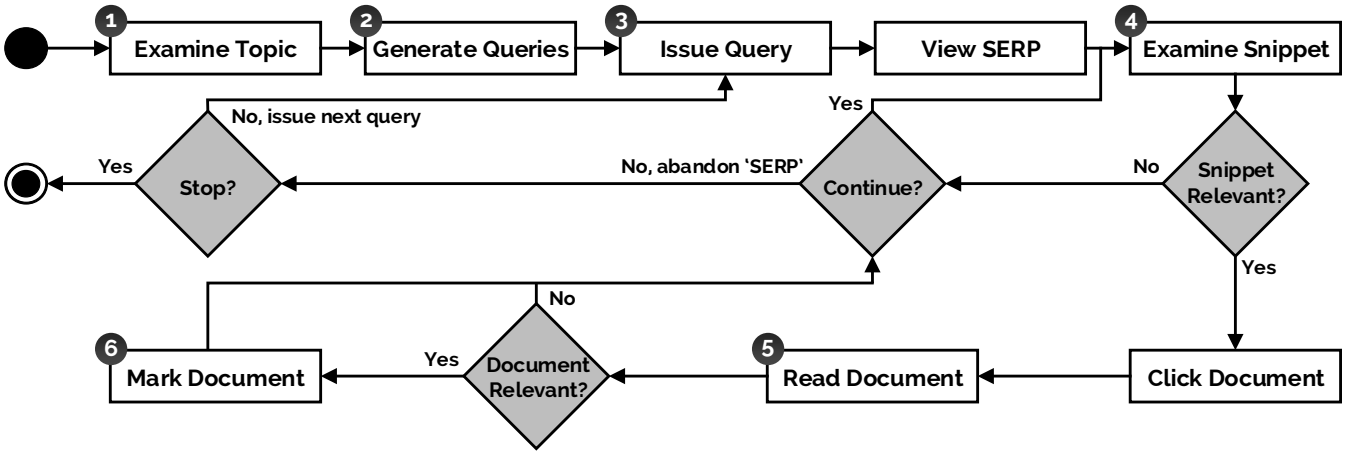

Figure 3: A flowchart of the *Complex Searcher Model (CSM)*, from Maxwell et al. [14] - presently used within SimIIR as the underlying searcher model. The flowchart illustrates the key components of the model (in white) and decisions (in grey) that simulated searchers consider. Numbers are associated with the steps as described in Section 2.9. The CSM is adapted from the works of Baskaya et al. [6] and Thomas et al. [19].

of different simulated query generation strategies exist, such as the approaches discussed by Baskaya [5] and Keskustalo et al. [11]. Queries also need not be generated - a list of queries issued by real-world searchers can be used, if desired. This functionality has been already implemented through a `PredeterminedQueryGenerator` class, which takes as input a list of queries to be issued. This is useful for comparing the performance of real-world searchers and simulated searchers under similar conditions, as used by Maxwell et al. [14].

In addition to the generator detailed above, a number of query generation strategies have been operationalised as SimIIR classes. Several of the querying strategies proposed by Baskaya [5] and Keskustalo et al. [11] have already been implemented, such as the `SingleTermQueryGenerator` that generates single term queries, the `BiTermQueryGenerator` that generates two term queries, and the `TriTermQueryGenerator` that generates three term queries. All of these classes generate queries from the provided topic title and description text. Work is also underway to incorporate terms from sources other than the topic title and description, such as previously examined snippets and documents. This will result in a wider potential vocabulary for query generation.

2.5 Snippet/Document Classifiers

We next describe another major component of the SimIIR framework - *classifiers*. These components are responsible for determining the attractiveness of a given snippet - or the relevancy of a document - to the provided topic that the simulated searcher is tasked to examine. Snippet and document classifiers can be specified individually to enable the simulator to take into account the fact that searchers may judge snippets differently from documents. As an example, a searcher may be more liberal when deciding the attractiveness of a snippet as opposed to a document's relevancy.

A series of snippet/document classifiers have already been implemented and are ready for use. These include: a *'TREC-style'* classifier, which judges everything to be relevant; an `InformedTrecTextClassifier` that uses TREC QRELS and probabilities (acting stochastically) to determine if a snippet/document should be considered attractive/relevant; and a `LMTextClassifier` which uses a language model (specified by optional configuration parameters) that acts deterministically to determine attractiveness and/or relevancy.

2.6 Stopping Strategies

The stopping strategy decides where a simulated searcher should stop examining a list of ranked documents. This concept is called *query stopping* [14], as opposed to *session stopping*, which is presently determined by the logger component (refer to Section 2.7). A variety of stopping strategies have been previously proposed in the literature, such as the *disgust* and *frustration point* rules [8, 12].

As two recent publications used the SimIIR framework to simulate the effects of stopping strategies [13, 14], a variety of strategies have already been implemented. As an example, these include: a `FixedDepthDecisionMaker`, implementing a naïve approach where simulated searchers will stop after examining x snippets regardless of relevance; a `NonrelDecisionMaker`, which operationalises the aforementioned disgust and frustration point rules [8, 12], where simulated searchers will stop after judging x snippets non relevant; and an `IftBasedDecisionMaker`, a stopping strategy which operationalises the implicit stopping rule encoded within *Information Foraging Theory (IFT)* [14, 16].

2.7 Loggers

A logger is the component of the SimIIR framework that determines the *costs* of interaction for the simulated searcher when performing actions, such as issuing a query. Interaction costs can be defined in the user configuration file. The logger therefore tracks and records all interactions that take place within a simulated search session, and is for example able to state when a predetermined time limit elapses - thus providing some form of session stopping. The details stored within the logger component can then be sent to the output controller component upon completion of the simulated search session to produce the `.log` output file.

To date, we have fully implemented a `FixedCostLogger` that considers each of the different actions undertaken by a simulated searcher, and as the name suggests, imposes a fixed cost upon each of them. Currently, a variable cost logger is in the early stages of development. This more ad-

vanced logger will for example provide the ability for SimIIR to impose variable costs that are dependent upon factors such as query or document length, for example [7, 18].

2.8 Search Contexts

The search context is the component of the SimIIR framework responsible for keeping a record of all issued queries, all examined snippets, documents (along with the judgements for each) and other interactions throughout a search session. The search context is in essence the 'memory' of the simulated searcher, and interacts closely with the specified logger component to record all events (refer to Section 2.7). By referring to the search context, a snippet classifier can for example determine what snippets the simulated searcher has previously seen, whether they were considered relevant or non-relevant, and what text was observed.

From the basic search context, we have derived a *relevance revision* search context, as used by Maxwell et al. [13, 14]. When considering a snippet relevant, a simulated searcher using the revised relevance search context can then revise its judgement of said snippet if the associated document is subsequently deemed to be not relevant. Such a technique can influence the stopping point of the simulated searcher if using a stopping strategy based upon the frustration point and disgust stopping rules [8, 12] (refer to Section 2.6) for instance. We envisage that as work in the area of IIR simulation progresses, more complex search contexts can be implemented. For example, one such approach could be a 'lossy' search context, where a simulated searcher would *forget* over time what snippets and documents have been examined.

2.9 The Searcher Model

When SimIIR is started, all the components are instantiated based upon the simulation and user configuration XML files. The process in which SimIIR simulates IIR is based upon the *Complex Searcher Model (CSM)*, a model of interaction proposed by Maxwell et al. [14]. The model, represented as a flowchart in Figure 3, is based upon the stochastic model presented by Baskaya et al. [6], but includes additional decision points as described by Thomas et al. [19]. While the CSM does not presently represent every aspect of the IIR process, it does provide a better representation than has been used previously. The model can be extended as research in this area progresses.

Essentially, the simulated searcher begins by ***(1)*** examining the given topic and title description. From the title and description, the simulated searcher then ***(2)*** generates a series of queries which are issued to the underlying search engine. The simulated searcher then ***(3)*** issues a query from the generated list, and then ***(4)*** proceeds to examine the first/next snippet in the ranked list provided. The simulated searcher can also decide to issue a new query, thus returning to *(3)*. If the snippet is considered relevant by the simulated searcher, ***(5)*** the document is then examined in full. If the document is also considered relevant, ***(6)*** the document is then marked relevant. If either the snippet or document are considered non-relevant, the simulated searcher then returns to *(4)* with the document unmarked.

3. SUMMARY

In this demonstration paper, we have described our new IIR simulation framework, SimIIR. The highly modularised architecture of the framework makes it straightforward to implement new simulation components, and will aid in pushing research forward in this area. Future work will see the development of more advanced components and the adaption of the CSM to facilitate this.

Acknowledgments We would like to thank Professor Kalervo Järvelin and Dr Heikki Keskustalo at the University of Tampere, Finland for their co-operation and participation in a STSM, funded by the ESF supported MUMIA Cost Action (reference `ECOST-STSM-IC1002-080914-049840`). The lead author is also financially supported by the EPSRC, grant number `1367507`. We would also like to thank Paul Thomas for his debugging efforts.

References

[1] L. Azzopardi. Query side evaluation: An empirical analysis of effectiveness and effort. In *Proceedings of the 32nd ACM SIGIR*, pages 556–563, 2009.

[2] L. Azzopardi. The economics in interactive information retrieval. In *Proc. 34th ACM SIGIR*, pages 15–24, 2011.

[3] L. Azzopardi, M. de Rijke, and K. Balog. Building simulated queries for known-item topics: An analysis using six european languages. In *Proc. 30th ACM SIGIR*, pages 455–462, 2007.

[4] L. Azzopardi, K. Järvelin, J. Kamps, and M.D. Smucker. Report on the sigir 2010 workshop on the simulation of interaction. *SIGIR Forum*, 44(2):35–47, 2011.

[5] F. Baskaya. *Simulating Search Sessions in Interactive Information Retrieval Evaluation*. PhD thesis, University of Tampere, School of Information Sciences, Finland, 2014.

[6] F. Baskaya, H. Keskustalo, and K. Järvelin. Modeling behavioral factors in interactive information retrieval. In *Proc. 22nd ACM CIKM*, pages 2297–2302, 2013.

[7] B. Carterette, A. Bah, and M. Zengin. Dynamic test collections for retrieval evaluation. In *Proc. 5th ACM ICTIR*, pages 91–100, 2015.

[8] W.S. Cooper. On selecting a measure of retrieval effectiveness part ii. implementation of the philosophy. *J. of the American Soc. for Info. Sci.*, 24(6):413–424, 1973.

[9] K. Järvelin. Interactive relevance feedback with graded relevance and sentence extraction: Simulated user experiments. In *Proc. 18th ACM CIKM*, pages 2053–2056, 2009.

[10] H. Keskustalo, K. Järvelin, and A. Pirkola. Evaluating the effectiveness of relevance feedback based on a user simulation model: Effects of a user scenario on cumulated gain value. *Information Retrieval*, 11(3):209–228, 2008.

[11] H. Keskustalo, K. Järvelin, A. Pirkola, T. Sharma, and M. Lykke. Test collection-based ir evaluation needs extension toward sessions — a case of extremely short queries. In *Proc. 5th AIRS*, pages 63–74, 2009.

[12] D.H. Kraft and T. Lee. Stopping rules and their effect on expected search length. *IPM*, 15(1):47 – 58, 1979.

[13] D. Maxwell, L. Azzopardi, K. Järvelin, and H. Keskustalo. An initial investigation into fixed and adaptive stopping strategies. In *Proc. 38th ACM SIGIR*, pages 903–906, 2015.

[14] D. Maxwell, L. Azzopardi, K. Järvelin, and H. Keskustalo. Searching and stopping: An analysis of stopping rules and strategies. In *Proc. 24th ACM CIKM*, pages 313–322, 2015.

[15] T. Pääkkönen, K. Järvelin, J. Kekäläinen, H. Keskustalo, F. Baskaya, D. Maxwell, and L. Azzopardi. Exploring behavioral dimensions in session effectiveness. In *Proc. 6th CLEF*, pages 178–189, 2015.

[16] P. Pirolli and S.K. Card. Information foraging. *Psychological Review*, 106:643–675, 1999.

[17] M.D. Smucker. An analysis of user strategies for examining and processing ranked lists of documents. In *Proc. of 5th HCIR*, 2011.

[18] M.D. Smucker and C.L.A. Clarke. Time-based calibration of effectiveness measures. In *Proc. 35th ACM SIGIR*, pages 95–104, 2012.

[19] P. Thomas, A. Moffat, P. Bailey, and F. Scholer. Modeling decision points in user search behavior. In *Proc. 5th IIiX*, pages 239–242, 2014.