Continuous Evaluation Framework for Information Retrieval Systems

Gabriela Gonzalez Saez

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Informatique
Unité de recherche : Laboratoire d'Informatique de Grenoble

Cadre d'évaluation continue pour les systèmes de recherche d'information

Continuous Evaluation Framework for Information Retrieval Systems

Présentée par :

Gabriela GONZALEZ SAEZ

Direction de thèse :

Philippe MULHEM CHARGE DE RECHERCHE HDR, CNRS DELEGATION ALPES	Directeur de thèse
Lorraine GOEURIOT MCF, UGA	Co-encadrante de thèse

Rapporteurs :

ERIC SAN JUAN
MAITRE DE CONFERENCES HDR, AVIGNON UNIVERSITE
JIAN-YUN NIE
FULL PROFESSOR, UNIVERSITE DE MONTREAL

Thèse soutenue publiquement le **3 octobre 2023**, devant le jury composé de :

PHILIPPE MULHEM CHARGE DE RECHERCHE HDR, CNRS DELEGATION ALPES	Directeur de thèse
ERIC SAN JUAN MAITRE DE CONFERENCES HDR, AVIGNON UNIVERSITE	Rapporteur
JIAN-YUN NIE FULL PROFESSOR, UNIVERSITE DE MONTREAL	Rapporteur
SIHEM AMER-YAHIA DIRECTRICE DE RECHERCHE, CNRS DELEGATION ALPES	Présidente
FLORINA PIROI SENIOR SCIENTIST, TECHNISCHE UNIVERSITÄT WIEN	Examinatrice
DIDIER SCHWAB PROFESSEUR DES UNIVERSITES, UNIVERSITE GRENOBLE ALPES	Examinateur

Invités :

LORRAINE GOEURIOT
MAITRE DE CONFERENCES, UNIVERSITE GRENOBLE ALPES
ROMAIN DEVEAUD
DOCTEUR EN SCIENCES, QWANT

Abstract

Classical evaluation of information retrieval (IR) systems uses a static test collection composed of a corpus of documents, a set of queries, and relevance judgments that indicate which documents are relevant to each query. In the case of Web search, the environment (e.g., the Web pages or the submitted queries) is continuously changing, and the hypothesis of using a static test collection is not representative of this changing reality. Moreover, new features are regularly added to the search engine creating a new IRS version. Finally, the changes in the search engine and the evolution of the test collection used to evaluate the system have an impact on the performance evaluation. To the best of our knowledge, there is no way to evaluate different IR systems using evolving test collections as a continuous evaluation of IR systems.

A continuous evaluation based on the classical evaluation paradigm should allow us to quantify the differences between evaluations. We call the differences between test collections a Knowledge delta (KD), and the performance differences between systems evaluated on these different test collections are called Result deltas (RD). Finally, the continuous evaluation is based on both Knowledge deltas and Result deltas. The related changes in both "deltas" will allow us to interpret the system's performances. We propose to create evolving test collections for a continuous result delta evaluation framework. An evolving test collection is a set of test collections that change across different epochs. It can be simulated using a static test collection, or it can be acquired using an evolving strategy. The evolving test collection is consequently used to evaluate systems in the continuous evaluation framework.

This work focuses on the computation of RD. With this goal, we propose a framework defined in three steps. First, we validate that two epochs are comparable using a set of reference systems that evaluate the comparability of test collection pairs. Second, we define the strategy to compare the performance of different systems across epochs. Here, we propose a comparison of systems using a reference system called pivot to create a continuous ranking of systems; a performance comparison in a specific epoch or interest relying on a set of reference systems to compute standardization and projection functions; and to define comparable grains across epochs. In the third and final step, the comparison strategy is applied to the tested systems, and longitudinal analysis is performed based on the computation of the RD according to each comparison strategy.

We test our evaluation framework using two simulated evolving test collections based on TREC-Robust and TREC-COVID, and we apply our framework using our acquired test collection LongEval. The results suggest that an evolving test collection provides more varied results than creating random shards from a test collection to evaluate performance variability. Using evolving test collections also show that there is a relation between the evolution of the test collections and the performance change of the systems. Finally, the continuous result delta evaluation framework relates the differences between the test collections and the differences in the performance of the systems to provide a

meaningful continuous evaluation, creating a correct ranking of systems, and a performance comparison across epochs. This work is the base of an evaluation framework that relates knowledge deltas with result deltas in an explainable continuous evaluation framework.

Keywords: Information Retrieval; Continuous Evaluation; Evolving Test Collection

Résumé

L'évaluation classique des systèmes de recherche d'informations (SRI) se fait à l'aide d'une collection de test statique composée d'un corpus de documents, d'un ensemble de requêtes et de jugements qui indiquent quels documents sont pertinents pour chaque requête. Dans le cas de la recherche d'information sur le Web, l'environnement (les pages Web, les requêtes soumises) change continuellement, et utiliser une collection de test statique ne tient pas compte de cette réalité changeante. De plus, de nouvelles fonctionnalités sont régulièrement ajoutées au moteur de recherche, créant de nouvelles versions du SRI. Ces modifications apportées au moteur de recherche, ainsi que l'évolution de la collection de tests utilisée pour évaluer le système, ont un impact sur l'évaluation des performances. À notre connaissance, la littérature ne propose aucun moyen d'évaluer différents SRI en utilisant des collections de tests évolutives comme support de l'évaluation continue de ces SRI. Une évaluation continue basée sur le paradigme d'évaluation classique devrait nous permettre de quantifier les différences entre les évaluations. Nous appelons les différences entre les collections de tests un "delta de connaissances" (KD), et les différences de performances entre les systèmes évalués sur ces différentes collections de tests sont appelées "delta de résultats" (RD). L'évaluation continue est basée à la fois sur les KD et les RD, qui nous permettent d'interpréter les performances du système. Nous proposons de créer des collections de test évolutives pour un cadre d'évaluation continue. Une collection de test évolutive est un ensemble de collections de test qui changent au fil des différentes époques. Elle peut être simulée à partir d'une collection de tests statique, ou acquise en utilisant une stratégie évolutive. Une telle collection de test évolutive est ensuite utilisée pour évaluer les systèmes dans le cadre de l'évaluation continue. Ce travail se concentre sur le calcul des RD. Dans cette optique, nous proposons un cadre défini en trois étapes. Tout d'abord, nous validons que deux époques sont comparables en utilisant un ensemble de systèmes de référence qui évaluent la comparabilité des paires de collections de tests. Ensuite, nous définissons la stratégie pour comparer les performances des différents systèmes à travers les époques. Nous proposons ici une comparaison des systèmes en utilisant : soit un système de référence appelé pivot pour créer un classement continu des systèmes, soit un ensemble de systèmes de référence pour calculer des fonctions de standardisation et de projection afin de définir des grains comparables à travers les époques. Dans la troisième et dernière étape, la stratégie de comparaison est appliquée aux

systèmes testés, et une analyse longitudinale est réalisée en fonction du calcul du RD selon chaque stratégie de comparaison. Nous avons testé notre cadre d'évaluation sur deux collections de test évolutives simulées basées sur TREC-Robust et TREC-COVID, et nous avons appliqué notre cadre en utilisant une collection de tests acquise à partir du Web, LongEval. Les résultats suggèrent qu'une collection de tests évolutive offre des résultats plus variés que la création aléatoire de fragments à partir d'une collection de test pour évaluer la variabilité des performances. L'utilisation de collections de test évolutives montre également qu'il existe une relation entre l'évolution des collections de test et le changement de performance des systèmes. Enfin, le cadre d'évaluation continue des RDs connecte les différences entre les collections de tests et les différences de performance des systèmes pour i) fournir une évaluation continue ayant du sens, et ii) créer un classement correct des systèmes et une comparaison des performances à travers les époques. Ce travail constitue la base d'un cadre d'évaluation qui relie les KDs aux RDs dans un cadre d'évaluation continue explicatif.

Mots clés: Recherche d'Information; Evaluation Continue; Test de Données Evolutif.

Acknowledgments

I want to express my gratitude to my supervisors Lorraine Goeuriot and Philippe Mulhem for the trust they placed in me throughout the course of this thesis. I appreciate their commitment and constant dedication to this research. I am grateful for their teachings and wise advice, both regarding the development of the thesis and my growth as a researcher. I finish this work feeling proud of the team we formed and everything I learned from you.

I would like to thank the School and Institute of *Ingeniería Civil en Informática* at the *Universidad Austral de Chile*. Thank you for making the school feel like home and for supporting me in all the projects I proposed. Undoubtedly, my decision to pursue an academic path has been influenced by all of you.

Thanks to my mom and my siblings Maca and Carlos for teaching me from a young age that I had to forge an independent path and that I am a strong woman capable of setting goals beyond what is established. Thank you for always motivating and supporting me.

Lastly, I want to express my gratitude to my husband Daniel and my son Sebastián. Daniel, thank you for motivating me to embark on this path and for accompanying me on it. Sebastián, I appreciate and admire your greatness in overcoming all the challenges that moving to a new country presented for you. Family, thank you for turning this experience into an enriching and love-filled family adventure.

Agradecimientos

Quisiera expresar mi agradecimiento a mis supervisores Philippe Mulhem y Lorraine Goeuriot por la confianza que depositaron en mí durante el transcurso de la tesis. Agradezco su compromiso y su constante dedicación con esta investigación. También agradezco sus enseñanzas y sabios consejos, tanto en relación al desarrollo de la tesis como a mi desarrollo como investigadora. Termino este trabajo sintiéndome muy orgullosa del equipo que formamos y de todo lo que aprendí de ustedes.

Quisiera agradecer a la Escuela e Instituto de Ingeniería Civil en Informática de la Universidad Austral de Chile. Gracias por hacer de la escuela mi casa y por apoyarme siempre en todos los proyectos que les propuse. Sin duda, mi decisión de continuar por un camino académico está influenciada por todos ustedes.

Gracias a mi mamá y mis hermanos Carlos y Maca por enseñarme desde pequeña que debía construir un camino independiente y que soy una mujer fuerte capaz de establecer metas más allá de lo establecido. Gracias por motivarme y apoyarme siempre.

Finalmente, quiero agradecer a mi marido Daniel y a mi hijo Sebastián. Gracias, Daniel, por motivarme a tomar este camino y por acompañarme en él. A Sebastián, agradezco y admiro su grandeza al enfrentar todos los desafíos que cambiarse de país implicó para él. Familia, gracias por hacer de esta experiencia una aventura familiar enriquecedora y llena de amor.

Acknowledgments of funding

This work was supported by the ANR Kodicare bi-lateral project, grant ANR-19-CE23-0029 of the French Agence Nationale de la Recherche, and by the Austrian Science Fund FWF, grant I4471-N.

Contents

Chapter 1

Introduction

Information Retrieval (IR) [118] aims at accessing relevant documents answering user's needs. The quality of an IR system is mainly related to its capacity to retrieve relevant documents. Since 1992, several classical evaluation campaigns have proposed reusable test collection to evaluate information retrieval systems involving a large number of scientists. The main ones are: the Text REtrieval Conference (TREC)[1], the Conference and Labs of the Evaluation Forum (CLEF)[2], and the Test Collection for IR Systems Project (NTCIR)[3], the Initiative for the Evaluation of XML Retrieval (INEX)[4], and the Forum for Information Retrieval Evaluation (FIRE)[5].

Classical evaluation of information retrieval systems evaluates a system in a *static test collection*, composed of a set of documents, a set of queries, and relevance judgments. The evaluation is conducted to test and measure the performance of a specific number of systems using a specific test collection. Any change in the evaluation, such as the document set, the queries set, the relevance judgments, or the evaluated systems, has an impact on the performance measurement [45, 133].

Thirty years of evaluation campaigns have led to the creation of many test collections. Overall, these evaluation campaigns share the same objective: comparing IR systems according to the exact same environment. Such evaluations are of great interest for the improvement of IR systems, but they are somewhat artificial: in real life (in any of industrial settings), elements (documents, queries, users, systems, ...) evolve.

In the case of Web search, the environment is continuously changing. The Web documents[6], as well as the users needs[7] are constantly evolving. Therefore the hypothesis

[1] https://trec.nist.gov/
[2] http://www.clef-initiative.eu/
[3] http://research.nii.ac.jp/ntcir/index-en.html
[4] https://inex.mmci.uni-saarland.de/
[5] http://fire.irsi.res.in/fire/2020/home
[6] https://www.worldwidewebsize.com/
[7] https://trends.google.com

of using a static test collection is not representative of this evolving reality, and it is an illusion to believe that a static collection is intended to work an industrial IR system. Therefore, offline evaluation of Information retrieval systems cannot be directly applied to live search systems in use in the industry: running one campaign a year is not enough; constantly tuning a system to a benchmark leads to overfitting the benchmark rather than improving the system's performance that is valid for different test collections.

We are aware that other ways exist to evaluate Web search engines, including, for instance, A/B testing [67]. We believe grounding our proposal on offline evaluation helps us compare to classical TREC evaluations. The goal and the resources needed to evaluate using online and offline evaluation paradigms differ: while the online evaluation seeks to measure the user feeling (satisfaction, preference, etc) from interaction behaviors (number of clicks, dwell time, etc), the offline paradigm focuses solely on the relevance retrieved rate comparing the system results against expert relevance judgments using a defined test collection. User-based evaluation schemes can give more realistic insights into the real system quality [34], but their results may not be as reusable as the offline measurements [106]. This reusability advantage makes the offline evaluation paradigm more suitable for comparing different systems that were evaluated in different evaluation environments. Additionally, it is important to be able to characterize the differences between the evaluation environments. This characterization is more suitable in the offline paradigm, as the collection of documents, queries, and judgments is known and modifiable.

There is a need for a framework to properly and continuously evaluate search systems. Compared to classical IR evaluation on static test collections, the continuous evaluation of IR systems integrates "time" into the evaluation process. Along with others, a longitudinal evaluation of an IR system is dedicated to checking if the quality of one (or several) system (s) works is higher, as good or worse, as time goes by. More precisely, the longitudinal evaluation is able to pinpoint some of the reasons for a system's behavior change.

In concrete, we exemplify our motivation in the following use case (Figure 1.1). Suppose that the search engine Qwant is constantly implementing modifications in its information retrieval model (exemplified in the logo changes that had been implemented over the years). At each modification, Qwant evaluates the system with a representative sample of its environment, which evolves constantly (represented by the changing Web pages set at the right of the figure). With the goal of decide if the search engine has improved, we aim at comparing to versions of the search engine, which were evaluated using different test collections. In such cases, the need for continuous evaluation emerges and has to be tackled.

We address the following research question: *How to compare the performance of different information retrieval systems evaluated when the test collections change?*

Figure 1.1: Qwant evaluation case with the IR system and Web pages changing across time.

1.1 Kodicare Context

This work was carried out within the Kodicare project (ANR-19-CE23-0029). Kodicare is a collaborative research project in an international context (PRCI) from the French *Agence National de Recherche* (ANR). Our partners are: from the academic perspective the Research Studios Austria (RSA), and from the industrial perspective the QWANT french company that proposes a Web search engine.

The general aim of Kodicare is *to provide a novel, sound, theoretical basis for unbiased continuous evaluation of IR methods, based on a quantification of the significant differences between benchmarks, on which future search systems in the real world can rely upon.*

Kodicare aims at adapting the classical information retrieval evaluation paradigm to a continuous evaluation one. The classical evaluation provides controlled test collection (i.e., set of topics, corpus of documents and relevant assessments) as a stable and meaningful environment that guarantees the reproducibility and explainability of system results. A continuous test collection needs to quantify the differences between the test collection elements (called *Knowledge delta*) and to study how such Knowledge deltas are related to the changes in IR systems performances, called *Result delta*. Kodicare is also concerned with bringing the academic results to scale though the collaboration with Qwant.

Within Kodicare, the objective of this thesis is to build a continuous evaluation framework for Information Retrieval Systems, focusing on the *Result delta* point of view.

1.2 Problem description

We address the evaluation of IR systems in the context of Web search. As stated before, the documents, queries, and users are constantly evolving. And also, the search engine is constantly implementing modifications in the IR model. To evaluate an IR system in this Web context, it is required to have a test collection that is modified across time as a real Web search environment. Here the first problem arises:

- There is no test collection with controlled evolutions of the set of documents, queries, or relevance judgments available in the literature which we can apply and test a continuous evaluation of information retrieval systems.

The second problem targets the fact that, at each evaluation attempt, different systems can be evaluated, needing to compare different systems evaluated on different test collections:

- We cannot compare the performance of a system evaluated in one test collection with a system evaluated in a second test collection because the performance variations are dependent on those changes. There is no evaluation framework that allows the comparison of systems evaluated in evolving test collections that also explains how the changes in the evaluation environment affect the performance results.

In the next section, we describe how this research work proposes to solve these problems.

1.3 Contributions

This thesis aims at developing a continuous evaluation framework of a real search engine of Web content, making possible the computation of result deltas between systems over different evaluation attempts. The contributions of this thesis are the following:

- A simulation method to create **evolving test collections from existing static test collections**. We propose a method to mimic the web environment by creating several sub-collections from an existing one. The simulation is defined by a set of features that controls the evolution across the sub-collections.

- The creation of **LongEval evolving test collections** from QWANT search engine. We propose an acquisition framework to gather documents, queries, and assessments periodically using an industrial search engine. LongEval was shared with the IR community as a CLEF evaluation lab.

- A **continuous evaluation framework** that is based on three steps: a comparability validation step, a comparison strategy step, and a longitudinal analysis. The framework allows the computation of the result deltas as the performance difference of systems evaluated in different epochs in an evolving test collection.

1.4 Report Structure

This thesis is organized into five chapters. They are organized as follows:

Chapter 2. Presents the evaluation of information retrieval systems. We start by presenting the information retrieval task and the Cranfield paradigm used to evaluate IR systems based on test collections. Then, we present different performance metrics from a query or system perspective. We focus on the limits of the current evaluation methodologies and explore the impact of changes in the test collection on the performance evaluation. We finish with a description of several test collections that could be used for a continuous evaluation of systems.

Chapter 3. Defines evolving test collections and proposes two methods to create them. The first method simulates an evolving test collection by using a static test collection, and the second method proposes a periodical acquisition of documents, queries, and assessments from search engine data. We compare and evaluate the stability of several simulated evolving test collections based on time features versus other state-of-the-art configurations. Finally, we present a real evolving test collection acquired in the context of CLEF LongEval shared task with the QWANT search engine.

Chapter 4. Describes our proposed Continuous Evaluation Framework. It describes the procedure to compute the performance difference between systems evaluated in different epochs of an evolving test collection. The framework is composed of three steps (i) a comparability validation step, (ii) a comparison strategy step, and (iii) a longitudinal analysis step. We test our framework using simulated ETCs and present its application using LongEval.

Chapter 5. Summarizes the contributions of this thesis, presents the main conclusions, and discusses possible future lines of study from this research work in the context of continuous evaluation of IR systems.

Appendix A. Presents our visualization tool that allows an exploratory analysis in the context of a continuous evaluation.

Appendix B. Presents a summary of this work in French.

Chapter 2

Information Retrieval and Evaluation

This chapter introduces an overview of evaluation methodologies in the information retrieval field. We present a general description of the information retrieval task in section 2.1; the offline evaluation method to measure the performance of a system and compare several systems using a common test collection is detailed in section 2.2; in section 2.3, we analyse the limits of the offline evaluation by showing the influence of the changes in the test collections on the performance measurement of information retrieval systems. Finally, we present the current effort to evaluate systems involving changing test collections in section 2.4 and some examples of dynamic test collections in section 2.5.

2.1 Information Retrieval

We take inspiration of Manning [68] to define Information Retrieval as:

> **Definition 1** ***Information Retrieval*** *(IR) is about finding relevant documents for a specific user's information need from a corpus of documents.*

An **Information Retrieval System** (IRS) is a computational system that a user can access to solve an *information need*. Baeza-Yates et al., [12] define that *The primary goal of an IRS is to retrieve all the documents which are relevant to a user query while retrieving as few non-relevant documents as possible.*

A key element in the definition of IR is the concept of *Information need*, which we define as:

> **Definition 2** ***Information need*** *is a request of a user to the IR system, usually expressed as a query.*

The **Query** expressed by the user is submitted to the IRS. It can be a list of words, an image, or an audio string, among others. Then, the IRS aims at returning a list of

documents that contain *relevant* information to the user's query. A **document** is the unit of information that the system retrieves. It can be a text, a video, an image, an audio, etc. In the case of a Web search engine, a document is a Web page. We call the **Corpus of documents** the set of documents that the IRS has access to.

The concept of **relevance** defines whether a *document* is related to the *information need*. The measure of relevance is subjective and widely discussed in the literature [16, 38, 74, 110]. Considering that the concept of relevance could be defined in several dimensions [75], we simply define it according to the relation between the user's information need and the retrieved document:

Definition 3 ***relevance*** *is the measure by which a document solves an information need.*

Figure 2.1 presents the Information Retrieval task as the process of retrieving relevant documents for a query. The IR task starts with a user that submits a **query** into an IRS. First, the IRS processes the query. Then, it computes a retrieval score to estimate the document's relevance to the query from the **corpus of documents**. Finally, the IRS returns a **ranked list of documents** according to the scores. These retrieved documents are ranked by decreasing retrieval scores. Ultimately, the user interacts with the ranked list to find relevant documents that solve her information need.

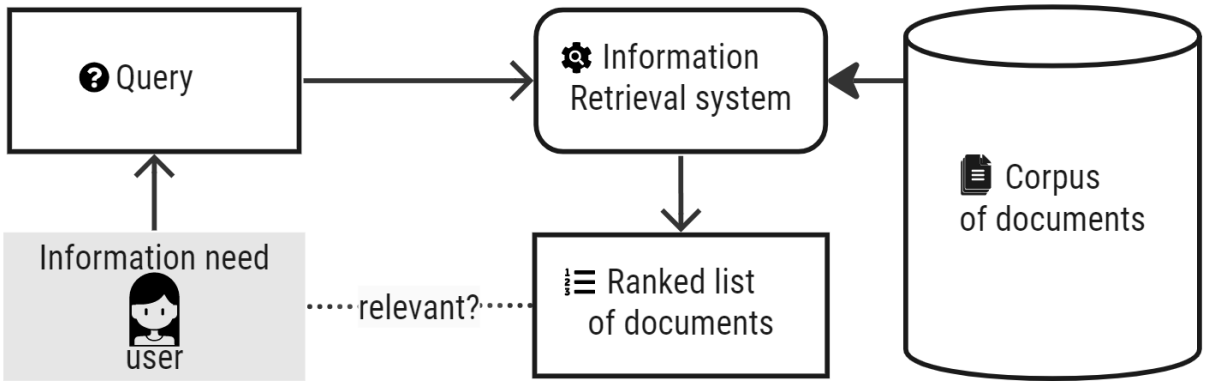


Figure 2.1: General description of the information retrieval task performed by a system.

The evaluation of Information retrieval systems is a challenging topic. One core goal of IR is providing documents that solve a user's information need. The query submitted by the user is usually incomplete or underspecified information [106]. The relevance of each document depends on the user's understanding of the document and her specific information need. This relevance measure is not explicitly informed to the IRS.

2.2 Information Retrieval Evaluation

The evaluation of information retrieval systems is dedicated to measuring the quality of a specific IRS. To do so, it is necessary to evaluate if the documents in the retrieved list are relevant to the user's information needs. In this section, we present the so called offline evaluation paradigm and the strategies to reproduce human relevance assessments of the IRS retrieved documents. First, we present the test collection in section 2.2.1. Then, a list of performance metrics which measures the performance of a system at the query or at a global level, in section 2.2.2. Finally, In section 2.2.3, we present methods to compare different systems evaluated in the same test collection.

2.2.1 Classical Offline Evaluation

The classical approach to measuring the quality of IR systems involves an offline evaluation, assessing IR systems in a controlled environment independent of real-time interactions. This evaluation method involves the utilization of pre-collected test collections to measure the performance of IR systems.

The offline evaluation of IR systems follows the well-established Cranfield paradigm [68], which entails a standardized methodology for evaluating IR system performance. This paradigm relies on reusable test collections and performance metrics to facilitate accurate and consistent evaluations. As described by Robertson et al. [91], a test collection, according to the Cranfield Paradigm, can be defined as:

- A set of **Topics and Queries**. A topic describes the information need using one or several descriptors as the user's context. The query is the expression of the topic, as it would be submitted to an IRS;
- A **Corpus**, the set of documents, as information units, from where the IRS finds results to the information need;
- A set of **Relevance Judgements** as an assessment that defines the relevance of a document to a specific topic, it simulates the user's judgement. This set is called *Qrels*.

As the topics and queries, documents in the corpus, and relevance judgments are not meant to change, we define this test collection as a static test collection:

A **Test Collection** (TC) is the tuple of three components (sets): documents (D), queries (Q) and relevance judgments ($Qrel$), defined by a set of assessment values (AV).

$$
\begin{aligned}
TC &= (D, Q, Qrel) \\
&\text{where,} \\
D &= \{d_j\}_{1 \leq j \leq nd}, |D| = nd; \\
Q &= \{q_k\}_{1 \leq k \leq nq}, |Q| = nq; \\
Qrel &= \{(d_j, q_k, f(d_j, q_k)\}, |Qrel| = nd \times nq; \\
&\text{with :} \\
f &: D \times Q \rightarrow AV \\
AV &= \{a_v\} \\
nd, nq &\in \mathbf{N}; a_v \in \mathbf{N}^+, \forall a_v \in AV
\end{aligned}
\tag{2.1}
$$

with AV, the domain that lists the set of assessment values. For instance, $AV = \{0, 1\}$ for binary assessments, where 1 represents relevant, and 0 represents documents non-relevant.

To evaluate an IRS using a test collection, the IR system retrieves, from the corpus, a ranked list of documents for each query of the test collection as the system result, called a *run* [106]. Then, the system's quality is evaluated using a performance metric that compares a run and $Qrel$. This evaluation process is exemplified in figure 2.2: first, a set of queries is submitted to the IRS; then, the IRS retrieves a ranked list of documents for each query (i.e., run). Finally, the relevance judgments (i.e., Qrel) of the test collection are compared against the run. In section 2.2.2, we describe how the performance metrics are computed by comparing the qrels versus a run.

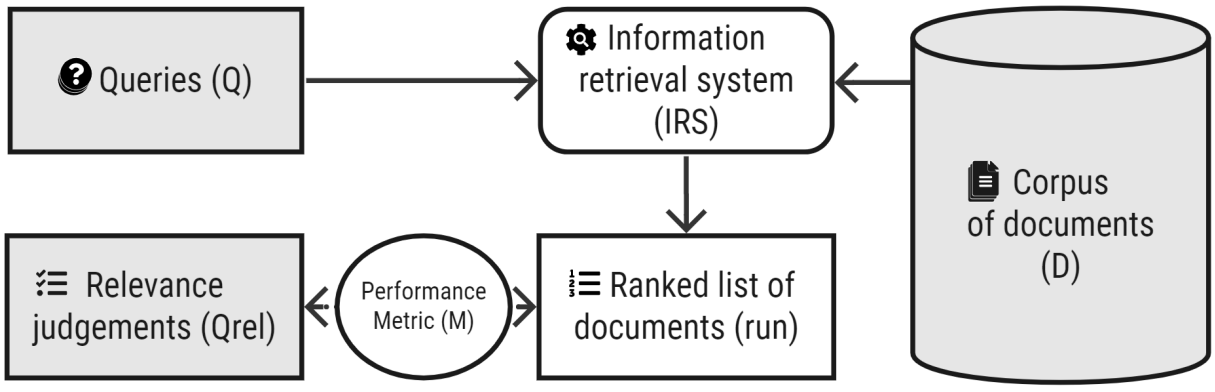


Figure 2.2: Information Retrieval Evaluation task using a Test Collection (components in grey).

When comparing the IR task (in figure 2.1) and the IR evaluation task (shown in

figure 2.2), we see that the evaluation query takes the place of the user's query. Similarly, the relevance judgment is a substitute for the user's interaction with the ranked list of documents by providing a relevance assessment value for each query-document pair. Notice that, in such a test collection, it is usual to find the topic that describes the context of the user when submitting the query as a narrative of the user's information need. However, the evaluation is performed with the query and without using the topic.

The construction of a test collection is usually related to a specific retrieval domain and use case [92] that guides the gathering of topics, queries, and documents. For example, the test collection may be focused on evaluating systems in a specific domain, as TRECVID [11] test collection that is oriented to the information video retrieval task, CLEFeHealth2020 [52] is oriented to the health search of a layperson; TREC-Robust [130] looks for poor performing queries on news documents; TREC-WEB2009 [35] test collection is oriented to the Web search. Other evaluation tasks focus on a use case and intend to study specific aspects of a search task. For example, in 1999, the TREC-Query track [21] created a test collection to analyze the difference in the retrieved documents using different queries for the same information need. Another test collection, UQV100 [13], focuses on the creation of multiple queries that different users would use to express the same information need.

Clough and Sanderson [37, 42] discuss the construction and use of test collections for evaluating the IRS and provide an overview of how queries, documents and relevant judgements can be gathered. Even when the specific details will depend on the requirements of the evaluation [107], the following steps provide a general framework for creating test collections:

- **Select topics and queries**: a set of topics is defined to reflect real information needs. It is possible that an expert creates a topic and one or several queries to describe a specific search case (as in CLEFeHealth2020 [52]). Another strategy is to extract real queries from a search engine log (as in TREC-WEB2009 [35]) to infer the context later in order to describe a narrative for the topic. Finally, another example is to define topics from social network posts (as in CLEF-MCM [51]). This process results in a list of queries in Q of the TC.

- **Gather documents**: The goal is to gather documents that are likely relevant to the queries. It should be large enough to ensure that relevant documents are included. The documents could be sampled from a specific system, as in MS-MARCO, where the documents are extracted from real Web documents in the Bing Index [80]; the corpus can be extracted using Web crawl, as CLEFeHealth2020 [52] that extract a defined set of URLs from the Common Crawl[1]; or using a defined dataset as CORD-19 dataset [134], the TREC-COVID [127] corpus, that selected documents based on

[1] http://commoncrawl.org

the presence of COVID-19 related terms the title or abstract of published scientific papers. This is the document corpus D in TC.

- **Obtaining relevance judgements**: The goal is to define which documents of the corpus D are actually relevant for each query in Q of the test collection. Since a topic may have multiple queries, the evaluation is performed for each individual query. The assessment could be performed in all pairs of document-query (as the initial Cranfield test collection [36]) or using a strategy to select which documents to assess for each query. TREC test collections (e.g., TRECVID [11]) propose to ask experts to judge the relevance of each document-query pair. Some studies propose to use click-logs from user behavior to extract the relevance of the documents implicitly [58, 64]. Another way to define the relevance of a document without human judgments (i.e., pseudo-relevance judgments) is to estimate the relevance from the structure of the documents [48] or from the relation between documents [7]. The final result of the assessment process is a tuple (d, q, av) that defines the relevance with an assessment value ($av \in AV$) for a query ($q \in Q$) and document ($d \in D$). The set of assessment values are finally the $Qrel$ set of the TC.

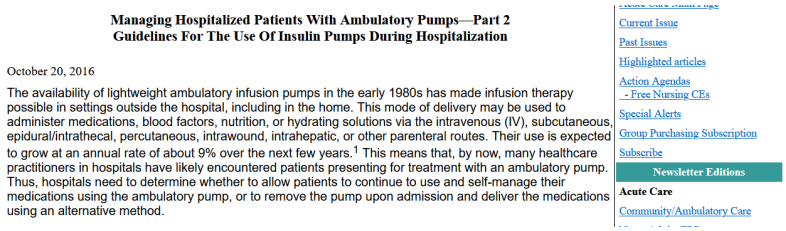
Text REtrieval Conference (TREC), Cross-Language Evaluation Forum (CLEF), NII Test Collection for IR Systems (NTCIR) and the Forum for Information Retrieval Evaluation (FIRE) are evaluation campaigns that propose a set of open challenges running every year with the goal of creating new test collections. In TREC evaluation campaigns, a group of organizers defines a specific information task to solve and then invites teams to submit the results of an IR system using a specific set of documents and queries following defined instructions [100]. To evaluate the system's results, the relevance judgment follows a pooling strategy [140], where the results of the systems are used to create a pool of documents that will be used to assess the relevance of the document-query pair. This selection strategy assumes that if any system retrieves a document, then it is probably not relevant to the query. Therefore, the assessment value for that document-query pair is the same value as the pairs judged as non-relevant.

Illustration with a use case: CLEF eHealth 2021 TC. We illustrate the creation of such test collection through the CLEF eHealth2021 test collection [119] created on the CLEF 2021 evaluation campaign. CLEF eHealth2021 has the general purpose of assisting laypeople in finding and understanding health information to make enlightened decisions. The test collection defines 100 topics created by medical experts and 50 topics extracted from social networks. An example of a topic of CLEF eHealth2021 is presented in listing 2.1. The $< id >$ tag is the topic identifier, the query is contained in the $< query >$ tag, and $< narrative >$ presents the context of the topic as a description of the user's information need.

Listing 2.1: Example topic from CLEF eHealth2021 test collection.

```
<topic>
    <id>8</id>
    <query>
        best apps daily activity exercise diabetes
    </query>
    <narrative> I'm a 15-year-old with diabetes. I'm planning
        to join the school hiking club. hat are the best apps
        to track my daily activity and exercises?
    </narrative>
</topic>
```

The corpus consists of 5 million medical Web pages from selected domains acquired from the Common Crawl. Figure 2.3 presents an example of a document of CLEF eHealth2021. The URLs were selected using Bing results for CLEF ehealth2018 and CLEF ehealth2020 queries.



Managing Hospitalized Patients With Ambulatory Pumps—Part 2
Guidelines For The Use Of Insulin Pumps During Hospitalization

October 20, 2016

The availability of lightweight ambulatory infusion pumps in the early 1980s has made infusion therapy possible in settings outside the hospital, including in the home. This mode of delivery may be used to administer medications, blood factors, nutrition, or hydrating solutions via the intravenous (IV), subcutaneous, epidural/intrathecal, percutaneous, intrawound, intrahepatic, or other parenteral routes. Their use is expected to grow at an annual rate of about 9% over the next few years.[1] This means that, by now, many healthcare practitioners in hospitals have likely encountered patients presenting for treatment with an ambulatory pump. Thus, hospitals need to determine whether to allow patients to continue to use and self-manage their medications using the ambulatory pump, or to remove the pump upon admission and deliver the medications using an alternative method.

Current Issue
Past Issues
Highlighted articles
Action Agendas
- Free Nursing CEs
Special Alerts
Group Purchasing Subscription
Subscribe
Newsletter Editions
Acute Care
Community/Ambulatory Care

Figure 2.3: Example of document in CLEFeHealth2021 corpus.

The relevance judgments were assessed by experts in the medical domain, considering three dimensions of relevance, *topicality*, *credibility* and *readability*. For example, following a TREC format[2], listing 2.2 presents the topicality relevance judgement for topic 1 (first column) with respect to four documents (third column), using a relevance value (in the fourth column) between 0 to 2:

Listing 2.2: Extract of qrels file from CLEFeHealth2021 test collection.

```
TOPIC    ITERATION    DOCUMENT                              RELEVANCE
1        0        a6195d99-f7d7-43ec-907c-435cb7a62ee7     2
1        0        80fd9af1-c9b6-4d82-b62a-fe16d5b9d76f     1
1        0        4a8c5d85-b2d0-43bc-83f4-c1acfe0b3481     2
1        0        22990a4b-6a9f-4e00-9e72-aef0fbee202b     0
```

[2] https://trec.nist.gov/data/qrels_eng/

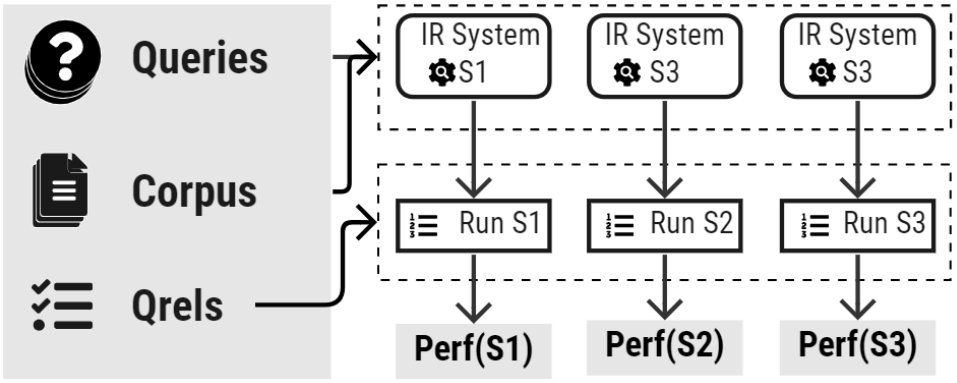


Figure 2.4: General description of IR evaluation task to compare the performance of several IRS using the same test collection.

Two other assessment files were generated, as the challenge also assessed the documents in terms of credibility (qcredibility) and readability (qreadability)[3].

In conclusion, the purpose of test collections is to evaluate an IRS by comparing its performance with other IR systems. Figure 2.4 presents an example of several systems using the same test collection to estimate their performance. All the systems evaluated using the same test collection can be compared. In the following section, we detail how the ranked list is compared to the set of assessments to compute the system's performance.

2.2.2 Performance Evaluation

Evaluating and comparing systems' performances is complex. Information Retrieval is an empirical task based on inferences made from the data: model specification, parameter estimation, and model evaluation [6]. To quantify the performance of an IR system, a test collection and evaluation metrics are needed [68]. It requires measuring the effectiveness of the system per query by comparing the ranked list of documents retrieved by the system against a set of pre-determined relevant documents (i.e., relevance judgments) using a defined set of metrics. Many evaluation metrics exist, measuring different facets of the performances of a system [109]. Classically, metrics are computed on the result list for one query and then averaged over a number of queries.

To compare the performance of two systems and determine whether system A outperforms system B, metric measures can be compared directly on a query-by-query basis, or as a global performance value for each system (such as the mean performance across queries), or as a distribution of performance values. In this way, it is possible to determine whether the difference between the compared systems is statistically significant or not,

[3]Accessible at https://github.com/CLEFeHealth/CHS-2021/tree/main/assessments

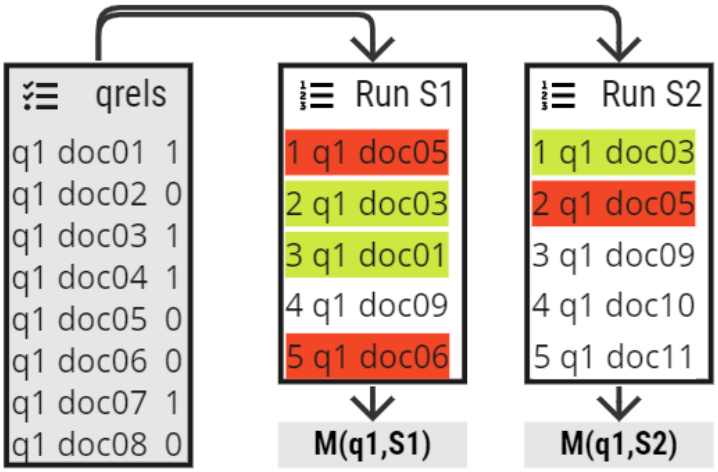


Figure 2.5: Results of system $S1$ (run S1) and $S2$ (run S2) using the relevance judgments (qrels) of query $q1$. Documents are in red for non-relevant ones, in green for relevant ones, and without color for unjudged documents.

according to a specific hypothesis test.

2.2.2.1 Per Query-based Metrics

The metrics measure a capacity of a system to retrieve relevant documents. In this section, we review the computation of performance metrics considering one query. Figure 2.5 illustrates how the $Qrel$ is compared to a run considering only one query (e.g., q_1). The first box shows which documents of the test collection are relevant (i.e., relevance value = 1); the second and third boxes show the result of systems S1 and S2 for query q1, in color are illustrated which retrieved documents are relevant (in green, doc03 and doc01), which are irrelevant (in red, doc05 and doc06), and which documents are not judged (without color, doc09, doc10, doc11).

The two main evaluation measures are the ***Recall*** and the ***Precision***:

$$Recall = \frac{|\ \{\text{ relevant documents retrieved }\}\ |}{|\ \{\text{relevant documents in qrels}\}\ |}$$

$$Precision = \frac{|\ \{\text{relevant retrieved documents}\}\ |}{|\ \{\text{retrieved documents}\}\ |}$$

The *Recall* metric measures the capacity of the system to retrieve **all** the relevant document, and the *Precision* metric evaluates the capacity of the system to retrieve **only** relevant documents. In our example (Figure 2.5):

$$Recall(q1, S1) = 2/4 \qquad Recall(q1, S2) = 1/4$$
$$Precision(q1, S1) = 2/5 \qquad Precision(q1, S2) = 1/5$$

the performance of S1 is higher than S2 for Recall and Precision.

Recall and *Precision* are complementary metrics (an ideal system should retrieve all and only relevant documents), then it is meaningful to mix them classically in the *F-measure* (harmonic mean of *Precision* and *Recall* values):

$$F\text{-}measure = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Recall and *Precision* are computed based on the full set of retrieved documents. Some metrics assume that a user does not review all the documents but until a specific point in the results. Recall at rank ($R@k$) and Precision at rank ($P@k$) metrics assume the user interacts with the first k documents in the ranked list.

$$R@k = \frac{|\{\text{relevant documents retrieved in first } k \text{ positions }\}|}{|\{\text{relevant documents in qrels}\}|}$$

$$P@k = \frac{|\{\text{ relevant documents retrieved in first } k \text{ positions }\}|}{k}$$

R-precision is the precision at rank R, where R is the total number of documents relevant to the query [8]. This metric allows examining a different number of documents per query, depending on the number of relevant judgements.

$$R\text{-}precision = \frac{|\{\text{ relevant documents retrieved in first } R \text{ positions }\}|}{R}$$
$$R : |\{\text{relevant documents in qrels}\}|$$

By definition, *R-precision* is the precision value at rank R ($P@R$), which is also the definition of recall at rank R ($R@R$) [10]:

$$R\text{-}precision = P@R = R@R$$

Reciprocal rank (RR) assumes the user stops when the first relevant document is found. It is used in Question-Answering search cases [87]:

$$RR = \frac{1}{k}$$
$$k : \text{ranking position of first relevant document}$$

If we apply the metrics to our illustrated example (Figure 2.5), if the user review only 2 documents, the system's performance is:

$$R@2(q1, S1) = 1/2 \qquad R@2(q1, S2) = 1/2$$
$$P@2(q1, S1) = 1/2 \qquad P@2(q1, S2) = 1/2$$

as we have 4 relevant documents, $R = 4$, one relevant document in the first four positions, and the first relevant document is in rank position two:

$$\textit{R-precision}(q1, S1) = 2/4 \qquad \textit{R-precision}(q1, S2) = 1/4$$
$$RR(q1, S1) = 1/2 \qquad RR(q1, S2) = 1/1$$

Therefore, S1 has the same performance as S2 according to $R@2$ and $P@2$, S1 has better performance than S2 according to *R-precision*, and the opposite occurs if the performance is measured with RR with $S2$ better than $S1$.

An extension of these metrics is Average Precision (AP), which takes the average of the precision at every document retrieved:

$$AP = \frac{1}{r} \sum_{k \in r} P@k$$

$$r : \{\text{position of all the relevant documents in the run}\}$$

Robertson [90] proposes a simple interpretation for this metric as the expected precision with the user's stopping point being uniformly distributed over all the relevant documents of the topic.

Widely used metrics, like AP and *R-precision*, require assessing all the documents of the retrieved set to obtain accurate values. However, in several cases, it is not possible (because of prohibitive time and cost) to have the complete set of documents judged for each query. The preference-based measure ($bpref$) [20] is inversely related to the fraction of judged non-relevant documents retrieved before relevant documents, making it robust against incomplete relevance information.

$$bpref = \frac{1}{R} \sum_{k \in r} 1 - \frac{|\{\text{non-relevant documents ranked higher than k}\}|}{min(R, |N|)}$$

$$N : \{\text{documents judged as non-relevant}\}$$

The idea behind $bpref$ is that for a user, any relevant document is preferred over any non-relevant document [12].

Another example of a metric robust to incomplete relevance assessments is inferred AP ($infAP$) [9], which infers a full set of assessments from a small fraction of judged documents: it uses estimates of average precision of multiple systems, together with an estimation of the total number of relevant documents in a query, computed using a small

number of relevance judgments.

The position of the documents in the run has not the same importance for the user. Ranked-based metrics are measures that consider the position of the documents in a result list. Some evaluation measures consider that the ranking quality is higher if the relevant documents are ranked first rather than at the bottom positions. The Discounted Cumulative Gain (DCG) [62] takes the ranking into account by penalizing with a discount value every next position in the ranking list.

$$DCG = \sum_{k=1}^{|\{run\}|} \frac{rel(d@k)}{log(k+1)}$$

$rel(d@k)$: relevance value of document in position k

As DCG values could vary strongly between queries, it is normalized against the ideal DCG value of the list ($IDCG$), leading to the normalized discounted cumulative gain ($nDCG$).

$$nDCG = \frac{DCG}{IDCG},$$

$IDCG$ is DCG defined on an ideal run that considers all relevant documents ordered by relevance value (the same number of documents as run).

DCG (and $nDCG$) are able to handle graded assessments (n-levels of relevance). Previous metrics, like AP, *R-precision*, and RR have to be adapted because they rely on binary values only (relevant with value 1 and non-relevant as value 0).

The Rank Biased Precision (RBP) [78] describes the users' persistence in stepping through each search ranking.

$$RBP = (1-p) \times \sum_{i=1}^{n} rel(d@i) * p^{i-1}$$

p : probability of view the next item i in the ranked list

The Expected Reciprocal Rank (ERR) [29] can be seen as an extension of the classical reciprocal rank to the graded relevance case. This metric is defined as the expected reciprocal time that the user will take to find a relevant document.

$$ERR = \sum_{k=1}^{K} \frac{1}{k} rel(d@k) \prod_{i=1}^{k-1} (1 - rel(d@i))$$

K : number of retrieved documents

ERR assumes a user examines documents by a cascade browsing model (in sequence). *ERR* quantifies the usefulness of a document at rank i conditioned on the degree of relevance of the items at ranks lower than i (i.e. appearing before i in the result list).

Many metrics could be (and are) used to define the performance of the systems: they all help addressing different aspects of the information retrieval evaluation. Computing the performance of S1 and S2 (figure 2.5), using some metrics, showed how considering different metrics we can conclude differently. S1 is better than S2 according to Recall, Precision and RPrecision, but S2 is better than S1 in terms of RR. In summary, table 2.1 presents the characteristics of the metrics described in this section. Most of the presented metrics need to fully examine the result list of documents to compute an evaluation taking into consideration the ranking of the documents. Only a few metrics penalize the position in the ranking and are defined to work with graded assessments.

		Characteristics			
		Full Set Examination	Ranked based	Penalization Score	Graded Assesment
Metrics	*Precision*	X			
	Recall	X			
	AP	X	X		
	P@k R@k	X(@k)	X		
	RR R-precision	X(@R)	X		
	DCG *nDCG*	X	X	X	X
	bpref		X		
	infAP		X		
	RBP	X	X	X	X
	ERR	X	X (Cascade)		X

Table 2.1: Characteristics of the metrics described in section 2.2.2.1.

2.2.2.2 Global System's Performance

The metrics above are computed for one query only. However, to estimate the quality of a system, several queries have to be considered, as one system may be very effective for one query and not for others. Increasing the number of queries considered is a way to increase the confidence in the results of a system evaluated using the test collection [129].

The most common evaluation metric considering several queries is the mean Average Precision (mAP), computed by the arithmetic mean of the average precision (AP) values for each individual query. More generally, it is also common to calculate the mean of a

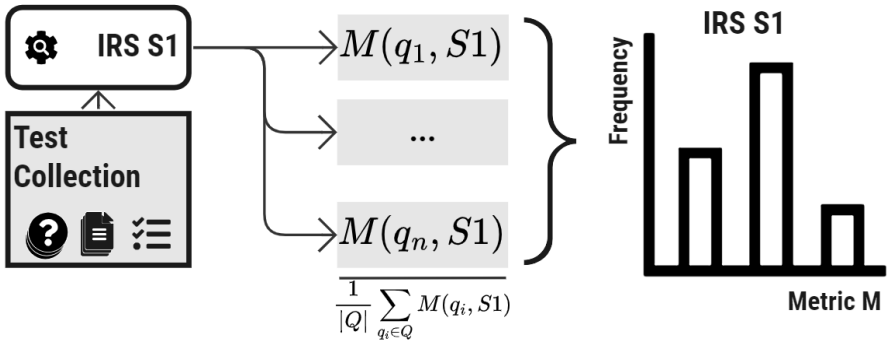


Figure 2.6: Performance result of an IRS (S1) using a Test Collection. (a) Illustrates the implementation of the IRS S1 using the test collection; (b) represents the per-query performance result and the overall metric; and (c) represents the histogram of the performance of IRS S1.

metric across the entire query set. However, some metrics are ordinal-scale only (e.g., RR and ERR), and the mean is not recommended to be computed [49, 101]. In this case, it is possible to compute the Median of the values as the overall performance of the system. The geometric mean of per-query metric measures is also used in the literature [130]. For example, $GMAP$ is used to highlight improvements for low-performing queries. To describe the performance of a system, other classical order statistics like variance or standard deviation can be computed.

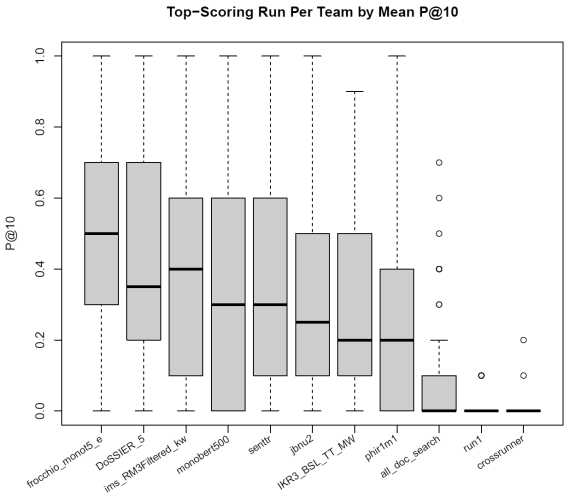
Another description of the systems' performance is to compute its score performance distribution using the performance of each query in the test collection. This approach considers that each query result is a unit of measurement, assuming that the queries were sampled from some population queries [93].

Figure 2.6 presents an example of the evaluation results for an IRS (S1) using a Test Collection. First, the performance is computed query by query. Then, an overall performance value might be computed considering all the queries (e.g., the mean performance value measured with M metric). Finally, the histogram of the per-query scores is used to describe the system's performance. The per-query performance scores can be further used to model a score distribution of the system [124] or to run statistical tests [98] with the goal of comparing systems, which is explained in section 2.2.3.

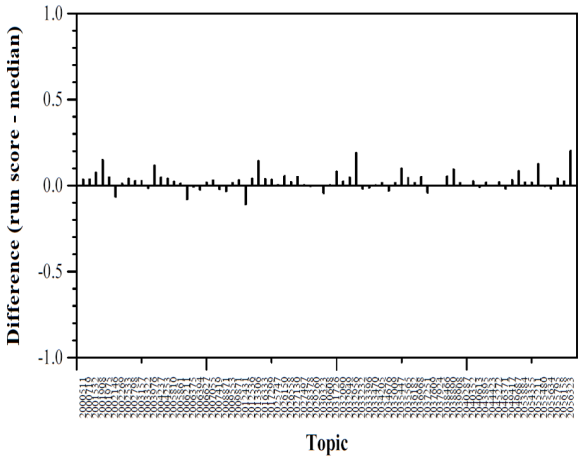
2.2.3 Comparing Systems using the same Test Collection

We have seen that testing a system involves using several metrics that measure different dimensions of the retrieval task. The question is: "How to assess if system S1 performs better than system S2?", in other words, if system S1 outperforms system S2?

The literature compares systems in a uni-dimensional manner, trying to find correla-



(a) Distribution of scores across topics for each team at Clinical Trials TREC 2022 [89]. Comparison of the best system for each of the 11 participating teams-



(b) Per-topic difference from median for Average Precision for Passage Ranking runs. UGA submission (unicoil_reranked[4]) at DeepLearning TREC 2022.

Figure 2.7: (a) Box-plot example. (b) Precision-histogram example.

tions and significant differences between systems. Visually, the most common tools are the box-plot [5, 69, 117] and average precision histograms [79]. These tools allow for comparing the retrieval performance of two or more systems through visual inspection. While a box-plot helps to compare the general distribution of the data and outliers, average precision histograms show the difference, per query, between a system and a set of other systems that are evaluated on the same queries. The performance difference between the system and the median average precision of other runs on that query is computed for each query. This histogram identifies the queries that are most challenging for an IRS [14].

Figure 2.7 presents an example of a box-plot and the average precision histogram for systems participating in a TREC evaluation campaign. We can compare the score distribution for any pair of systems using the box-plot in figure 2.7a. The comparison between systems is based on the performance of each system across the full set of queries (defined as *topic* at TREC DeepLearning Track). We can observe that the maximum quartiles are common for several systems, even if we can not interpret from the graphic which is the best-performing query. The best P@10 mean value in this example is 0.5 (*frocchio_monit5_e* system). In figure 2.7b, the median performance per topic (considering 100 systems) is compared to the performance of *unicoil_reranked* system on each topic. This system is then better than the median in 52 topics, and worse than the median in 18 topics, with the biggest negative difference in topic 2009871.

[4]reported in https://trec.nist.gov/pubs/trec31/appendices/deep/unicoil_reranked.pdf

In addition to comparing the performance measures, statistical tests may be computed. The goal of such a test is to detect significant improvements in the performance of IR systems. A significance test is based on the computation of a test statistic given a null hypothesis, a distribution of the statistic, and the computation of a significance level to determine the likelihood that the null hypothesis occurred in an experiment [17]. A low significance level allows rejection of the null hypothesis, but if it cannot be rejected, the observed difference may be due to evaluation noise [114].

The most widely used statistical tests in IR are Student's t-test, Wilcoxon signed-rank and bootstrap [24]. The null hypothesis of these tests is: systems S1 and S2 are random samples from the same normal distribution in Student's paired t-test; systems S1 and S2 are random samples from the same distribution in the bootstrap test; and that systems S1 and S2 have the same distribution for Wilcoxon rank test (the same hypothesis that Sign and Fisher's randomization tests) [72].

Smucker et al., [113, 114] analyse several statistical significance tests applied to IR and argue that Student's paired t, bootstrap and randomization tests agree with each other and arrive at the same conclusion regarding the statistical significance of their results, while Wilcoxon signed rank test, disagrees with the other test results and therefore is not recommended to be used by IR researchers.

Classical information retrieval evaluation defines how to compute the performance of IR systems and allows the comparison of two or more systems evaluated in the same test collection. Using the performance scores of the evaluated systems across several queries, it is possible to estimate its distribution. When evaluating statistical tests on the results of two systems, it is possible to compare if the difference between the systems is statistically significant (e.g., rejecting the null hypothesis that two systems are random samples from the same distribution in the bootstrap test). Finally, a ranking of systems (RoS) can be built using a decreasing order of the systems' scores. The RoS, along with the significance level between systems, is the final result of the Classical evaluation task.

2.3 Limits: Changes in the Test Collections

One limit of the Cranfield paradigm (see section 2.2.1) is that it does not support the analysis of the performance of systems in dynamic contexts, such as the Web. Assessing the quality of Web retrieval needs a repeated or continuous evaluation with incremental document collections [63]. However, one of the most important constraints of the Cranfield evaluation is that it uses a common test collection for all the systems in comparison. The documents, queries, and relevance judgments remain constant for all the systems evaluated. Any document or query modification may invalidate the assessment performed for the query-document pair, probably changing the judgment; removing or adding documents might change the system's performance in a specific query. Moreover,

modifying the set of queries can change the global performance of the system.

For example, adding or removing documents from a test collection has an impact on a system's *Recall* and *Precision*. Table 2.2 illustrates the impact in *Recall* and *Precision* considering three factors: (i) if the document is added or removed from the test collection, (ii) if the document is (or was) retrieved by the evaluated IRS; and (iii) if the document is relevant or irrelevant to the query. Combining these three factors, we notice that when a new relevant document is added to the test collection and the IRS retrieves it, the Recall and Precision might increase. However, if the added document is not relevant, the recall will be maintained because the set of relevant documents is not changed, but the precision will decrease if the IRS retrieves this document.

		adding a document, that the IRS		**removing** a document, that the IRS	
		retrieves	do not retrieve	retrieves	do not retrieve
document is:	**relevant**	*Recall* ↑ *Precision* ↑	*Recall* ↓	*Recall* ↓ *Precision* ↓	*Recall* ↑
	irrelevant	*Precision* ↓		*Precision* ↑	

Table 2.2: *Recall* and *Precision* performance change when a document is added or removed from a test collection. The document could be relevant or not relevant for a specific query, and it could be retrieved or not retrieved by the IRS. The ↑ represents that performance increase, and the ↓ symbol represents a decrease.

We present now three lines of work in analyzing performance change, first focused on the documents and judgments changes, then on changes in the set of topics, including query variations, and finally, on all these components together using Analysis of Variance.

2.3.1 Document Collection Stability

In this section, we present the works related to the analysis of the impact on the performance of a system if the set of documents is different.

Sanderson et. al. [108] simulate variations of a static test collection to analyse the relative performance of a retrieval system. They create sub-collections by splitting a test collection, doing so, they found substantial and statistically significant differences in the relative performance of retrieval systems evaluated in different sub-collections. In the same line, posterior works [43, 44, 133] continue using random splits of the test collection as sub-corpora or replicates of the test collection to evaluate the system in

variable collections with the aim of explaining the effect of each component of the test collection in the performance of a system. As shown in [108], evaluations conducted on different sub-collections (splits of the document corpus with the respective relevance assessments) lead to substantial and statistically significant differences in the relative performance of retrieval systems, independently from the number of relevant documents that are available in the sub-collections.

Sakai [97] proposed to use bootstrapping to evaluate IR metrics. While Sakai's experiments were based on queries, Zobel and Rashidi [141] have shown the experimental variability using bootstrapping techniques on the corpus of documents across different performance metrics. This work considers only random corpus splits, and they do not focus specifically, as we do here, on detecting when the same ranking of systems is achieved.

Recent work of Rashidi et al. [88] proposes a method to split the corpus of documents using Bootstrap and different splitting features to analyse the impact of such features in the experiments' predictivity. They control the test collection splits by a "meld factor" of one characteristic: document length, document source, and high/low rank of the document. The "meld factor" measures the level of difference between the splits. They show that a bigger difference between the collection (i.e, bigger meld factor) is related to a bigger performance difference in the performance of the systems. Still, this difference is particular to each characteristic and metric, impacting the performance of the systems differently. Therefore, they show that the predictivity of experiments may be limited, and some measures are less predictive than others. However, this work does not define thresholds upon which we can rely to define similar collections.

Sub Test Collection Creation: One important element of these analyses is the creation of sub-collections to study the stability of the systems. Creating document sub-collections is called *sharding* in Ferro's and Sanderson's works [108]. The process to create shards from a static test collection is described as follows: (a) define the size N_{sub} of the sub-collection, (b) select a sample of size N_{sub} from the full set of documents. In the case of Zobel's work [141], it is proposed to create shards with duplicate documents. In this manner, the size of each sub-collection is the same as the original test collection, with elements repeated inside each sub-collection. The process follows a bootstrapping method for selecting each sample. In both works, to compute the performance of a system in a shard (i.e., sample sub-collection), a system is implemented in the full test collection, and the list of documents in the result of the systems (i.e., run) is filtered out to leave only the documents sampled for the current shard. Table 2.3 summarizes the main features of the previous works and the methodology used to create the collections.

2.3.2 Topic Difficulty

Another limit of the Cranfield paradigm is that it does not support changes in the queries, even when a user would perform the retrieval task using several queries for the same

name	papers	Size	Duplicates	Feature-based
sharding	Ferro et al., [46] Sanderson et al., [108] Voorhees et al., [133]	Half Collection	no	no
bootstrap	Zobel et al., [141]	Full Collection	yes	no
Bootstrap predictivity	Rashidi et al., [88]	Half Collection	yes	yes

Table 2.3: Sub-collections summary.

information need. Penha et al. [84] shows that the IRS are not robust to query variations, with an effectiveness drop of 20% compared to an original query. In the same line, Alaofi et al. [3] question *where do queries come from?*, demonstrating that query variations lead to different search results depending on how users define their queries and presenting the existing gaps in the analysis of the source of query variations in IR.

Changing, removing, or adding a query will change the results of the system and, therefore, its performance because the difficulty of the sets of queries is different. Topic difficulty can be defined as the average effectiveness of a set of systems for a query [138]. It is calculated using the Average Average Precision (AAP) measure: the average AP of all the submitted runs for a given query: the higher the AAP, the easier the query [126]. IR Systems retrieve a different number of relevant documents for different queries. The topic difficulty focuses on analyze how is the performance of the systems when evaluating easy queries versus difficult queries.

Mizzaro et al. [77] first found that the system effectiveness in TREC is affected more by easy queries than by difficult queries. Considering a metric that summarizes the performance of a system from several queries, the general performance of the systems is higher in easy queries than for difficult queries, then a bad performance on an easy query will affect more the final performance of the system than a bad performance on a difficult query. Then, Mizzaro [76] proposes the metric NMAP, which normalizes the AP (NAP) value with respect to the difficulty of the topic using two principles: *the easy and difficult* (to describe the queries) and *the good and bad* (to describe the performance of the systems). The metric considers that if the system has a good effectiveness on a difficult query, the evaluation should increase a lot; good effectiveness on an easy topic should increase the evaluation of the system by a small amount. The metric rewards the system for returning good results on difficult queries. In the cases of bad performances, if the system has a bad effectiveness on an easy query, the system's effectiveness should decrease a lot, and a bad effectiveness on a difficult query decreases system evaluation by a small amount. In conclusion, the metric penalizes systems performing poorly with easy queries.

The work of Zampieri et al. [138] addresses the effects of topic difficulty on the IR Evaluation, finding that the topic difficulty is affected by the document corpora (there was a significant corpus-effect on query difficulty in all of the collections tested).

2.3.3 Effect of Corpus, Query and System on the Performances

As a global framework, Ferro et al. [47] performed an Analysis of Variance to understand the effect of the systems, topics and sub-corpus in the performance of a system.

Using the ANalysis Of Variance (ANOVA) model, [44] showed that changing the test collection (splits of the documents corpus) leads to varying system performances (inconsistently across metrics). In the same line, [45, 133] model the system effect and the test collection effect on the performance metrics as separated factors, they define ANOVA models and General Linear Mixed Models (GLMMs) to analyse systems performances over several test collections with the goal of improving the measurement accuracy of retrieval system performance by better modeling the noise present in test collection scores.

Such studies are not aimed at system comparison but rather at measuring the effect of the test collection on a given system performance. They provide a better understanding of the measurement of performances but do not allow to compare two systems that are evaluated using different test collections.

An IR evaluation is composed of several elements and their changes impact the performance of the IR system, specifically on the result of the evaluated metrics. In a real scenario, the elements that compose the test collection might be updated, creating evolving test collections and IRS systems with different performance results in each test collection version.

2.4 IR Evaluation involving changing test collections

To cope with the evaluation of IR systems in the case of changing test collections, the literature addresses three dimensions. First, it is proposed to add new elements into the test collection, with the goal of analyzing the performance variability of the systems in a more realistic set-up as a test collection maintenance, which is described in section 2.4.1. Section 2.4.2 focus on the analysis of different queries, first trying to find a *generalizable* set of queries and then considering each query's difficulty to compute the systems' performance (query section). And third, the interpretation of the results across test collections is also tackled from the reproducibility perspective and through meta-analysis, in section 2.4.3.

2.4.1 Test Collection Maintenance

Few studies abandoned the static requisite of the test collection and proposed methods to *maintain* the test collection to represent a test collection that allows modification in the set of documents or judgments. Maintaining a test collection is defined as the task of updating its components by injecting or removing documents, queries or judgments to extend the *usable* lifetime of the test collection [115].

Soboroff [115] addresses the need to create a realistic Web search setting to evaluate IRS. Their experiments use a changing and growing document collection with a fixed set of topics and relevant judgments. Soboroff shows that it is possible to compare the performance of systems from different versions of the test collection despite the decay in relevance data due to the changing document collection. According to the $bpref$ evaluation measure (described in section 2.2.2.1), the rankings of the systems (RoS) in different versions of the test collection (as each set of the growing document collection) are similar to the RoS of the initial version of the test collection, leading to assess that systems are comparable across these versions.

Tonon et al. [121] proposed to increase the judged documents according to new systems evaluated in the same test collection. They tackle the problem of the bias introduced by a system being contributing or not to the pool of judged documents, considering that the document corpus and set of topics do not change over time. They show that this process is necessary as the test collection construction penalizes systems that did not participate in the pooling that might be more effective by retrieving diverse results than systems that took part in the pool [137].

Hashemi et al. [59] merge both previous works and propose a method to update an existing test collection by injecting new judged documents to make a test collection reusable in dynamic contexts. In this case, they add judged documents from a second test collection to the first collection to create an extended collection. The experiments show that the extended collection is more reusable than the first version. A test collection is reusable if it fairly evaluates retrieved runs that did not contribute to the pools used to construct the test collection [19].

These works present the need of creating alternative methods to incorporate incremental test collections on the evaluation of IR systems, as the classical offline evaluation method is not properly describing the Web search environment. They are used to analyse the performance difference of the same system in the dynamic setup but do not compare systems across changing collections.

2.4.2 Topics Standardization

The heterogeneity of the topic set in a test collection could affect the evaluation of the systems: different levels of difficulty of the topics or different sub-topics sets. In this section we explain how topic standardization tackles this problem.

Score standardization is an evaluation method that reduces the impact of the topic's difficulty on the IR system's performance [99, 123, 136]. It consists in normalizing the performance score for a topic by its observed mean and standard deviation over a set of runs/systems [135]. Urbano et al. [123] showed that even when the RoS between raw and standardized scores is the same, the RoS using mean scores may differ considerably.

Using standardized scores, systems can be compared across different test collections without worrying about topic hardness or normalization [99].

Webber et al. [136] first proposed standardization through a non-linear transformation with the standard normal distribution, to enable inter-collection score comparisons. Under standardization, the difficulty of a query is directly estimated from the scores achieved by a sample of experimental systems, and parameters derived from these estimates are then used to normalize the scores both of the experimental systems and of future systems. Their standardization score is given by $\frac{x-\mu}{\sigma}$, where μ is the mean and σ is the standard deviation of the topic results. Then the score is mapped onto the unit interval using the cumulative distribution function (cdf).

Sakai [99] proposed a simple linear transformation of the standardized scores, given by $A \times \frac{x-\mu}{\sigma} + B$, where μ is the mean and σ is the standard deviation of the topic results, and A and B are constant parameters. The goal of the work is to use standardized scores to compare systems across different test collections without worrying about topic hardness or normalization.

Urbano et al. [123] proposed that a transformation based on the empirical distribution is the most appropriate choice for this kind of standardization. They also show that the proposal of Webber and Sakai are special cases of a general class of standardization, based on the assumption of a specific distribution for the per-topic score.

2.4.3 Multiple Experiments interpretation

The Cranfield paradigm considers the evaluation of systems in the same test collection. Nevertheless, to validate the improvement of a new IR system it shall be compared against one or several baselines in different test collections to prove the reproducibility of the system results. To do that, the system and the baselines have to be actually applied on each collection; otherwise, they cannot be compared.

We evaluate a system in several test collections to analyse the extent to which the results of an evaluation can be regenerated under similar, or the same, conditions. To reproduce the performance of an IR system, it is necessary to evaluate it over similar, but different, test collections. By example, [105] analyzes the reproducibility capability of a system when using two sub-corpora from the OpenWeb collection (The TREC Contextual Suggestion (CS) track). Considering the potential impact that different collections may have on the retrieval effectiveness, the paper focuses on studying the gap in effectiveness between the two test collections, and it shows that the systems using the first one, Open

Web, performed better (retrieve more relevant documents) that systems using the second, ClueWeb12 [111].

When reproducing a system, is therefore necessary to comapare the performance of the system in the evaluated test collections. Breuer [18] proposes a methodology and the use of specific metrics to measure the difference between two information retrieval experiments using the same systems, over several test collections (With the goal of reproducing the results of the original retrieval system). Considering a system 'a' and a baseline system 'b', the **Effect Ratio** (ER) [103] is the ratio between the mean improvement in the replicated experiment and the mean improvement in the original experiment of the performance of systems 'a' and 'b'. The **Delta Relative Improvement** (DeltaRI) computes the difference between the original and reproduced runs, measuring the improvement of the system 'a' versus 'b' in terms of relative scores. This work evaluates the statistical significance of the results with paired and unpaired Student t-test. In the same context, Sakai [102] proposes to use a two-sample test to compare the statistical significance difference for an IR system evaluated two test collections. In summary, Effect Ratio and DeltaRI metrics quantify how close the effects of an original versus a reproduced system are in two test collections comparing the performance of a system 'a' versus a baseline 'b'.

To extend the interpretation of the results of a system evaluated in several collections Soboroff propose the use of **meta-analysis** [116]. In this case, the goal is not to just compare the reproducibility of the system in different collections, but to extract an overall conclusion of the reproducibility of the performance of the system in several scenarios. Meta-analysis also rely on comparing the performance of system "a" versus a baseline "b" over multiple test collections, which includes different topics and corpora settings. Soboroff proposed the use of meta-analysis to create an interpretable evaluation of systems that were tested over multiple collections, where one baseline is compared to a treated system resulting in a Delta Measure over multiple collections, generating a final mean difference with a confidence interval from the treat and the baseline systems. This technique is strongly related to measuring the improvement across multiple test collections of a system with a specific modification that differentiates it from the baseline system.

Figure 2.8 compares the presented methodologies: while Effect Ratio and DeltaRI measure the reproducibility of the performance of a system in two experiments, a meta-analysis allows us to compare the performance of a system in more test collections and to extract an overall conclusion of the reproducibility of the system in several scenarios using the final "overall effect" metric.

These methodologies require evaluating all the systems using the same test collections. They are the first step in interpreting the performance of systems evaluated in changing test collections, but they are not applicable when different systems are evaluated in different collections.

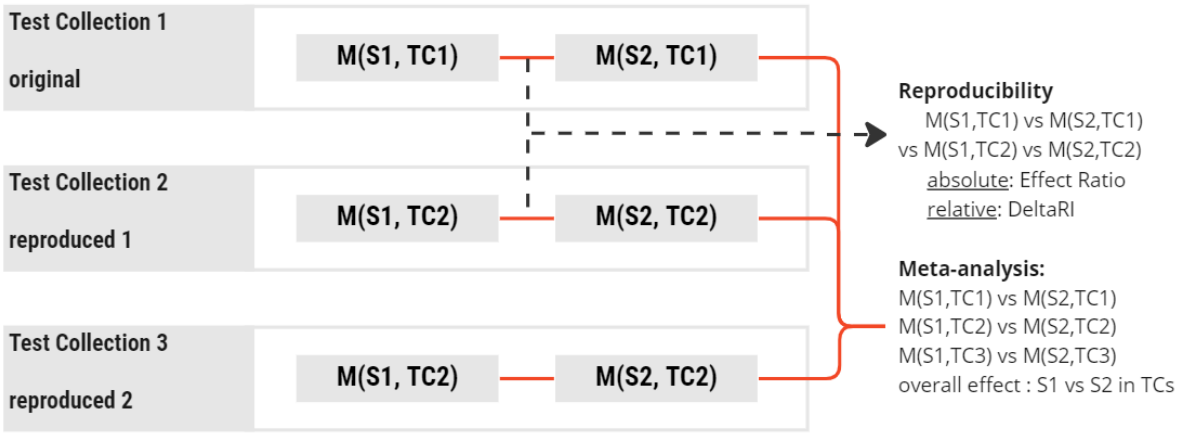


Figure 2.8: Comparison between Effect Ratio, DeltaRI and Meta-analysis.

2.5 Test Collections with changes

In this thesis, we focus on evaluating Web Information Retrieval Systems. Assessing the quality of Web search systems needs a repeated or continuous evaluation given incremental document collections [63]. We present the current test Collections that incorporate changes in their components, for example, test collections composed of different sets of documents, queries, or judgments. These datasets are essential to develop and test a **continuous evaluation of IR systems**. The main characteristic that we are looking into in these test collections is any information with respect to the acquisition of the documents, queries, and assessments, as the time factor should be included in a continuous evaluation process. We exclude Twitter-based collections due to the short length of the documents and temporal collections, as their use case is to detect events [23] outlying the general Web use case. We present below two test collections. TREC-COVID in section 2.5.1 and TREC-Robust in section 2.5.2.

2.5.1 TREC-COVID

TREC-COVID [127] is an example of test collection incorporating dynamic components, using a residual test collection strategy [104]. TREC-COVID has five rounds of evaluation with different corpus of documents and queries released at each round. The test collection of the current round is related to the previous one, sharing a set of overlapped documents and queries. At each round, all the relevant documents are removed, and five queries are added. TREC-COVID presents a dynamic test collection with emerging documents, topics, and relevance assessments at each round. However, as it is built as a residual collection, then the performance of the system could be degraded by the removal of relevant documents if we consider each round as a separate test collection.

Apart from the periodical acquisition of the rounds, each document in the test collection has a publication date that suits our time-based requirement. Therefore, the final version of TREC-COVID could be divided into several test collections to evaluate a system in the evolution of the Web during the COVID-19 pandemic.

Table 2.4 details each TREC-COVID round. All the resources are available on the web site[5], including the runs of the participants. The task is an approximation of continuous evaluation using offline test collections.

Round	CORD-19 release date	#Documents	#Topics	#Qrels	#Runs
1	April 10, 2020	51,078	30	30	143
2	May 1, 2020	59,887	35	35	136
3	May 19, 2020	128,492	40	40	79
4	June 19, 2020	158,274	45	45	72
5	July 16, 2020	192,509	50	50	126

Table 2.4: TREC-COVID rounds.

2.5.2 TREC-ROBUST

Robust Track 2004 looks to improve the consistency of retrieval technology by focusing on poorly performing topics [132]. The corpus of documents is the TREC Disks 4 and 5, excluding the Congressional Record subcollection. It is the same corpus as T07 and T08 because the Robust track uses their topics, plus the T06 topics give a total of 200 old topics. The task proposes 50 topics identified as Hard and, additionally, 49 new topics. The relevance assessment is made only on the 49 new topics and the evaluation is made separately for the old topics (200), the new topics (49), the hard topics (50), and all the sets combined.

Track	link	Corpus	Topics	Qrels	Runs
Robust 2004	trec.nist.gov/data/robust.html	† 528155 docs TREC Disks 4 and 5 (Congressional Record excluded)	250	250	† 110

Table 2.5: TREC-Robust test collections. † indicates resource under request or not available.

The TREC-Robust advantage is the creation date availability on each test collection document. Nevertheless, the queries and relevance judgments are gathered at one time.

[5] https://ir.nist.gov/covidSubmit/data.html

As in TREC-COVID, using this time feature, it is possible to create temporal splits of TREC-Robust to analyze how the performance of systems may change.

One of the most important constraints of the Cranfield evaluation is the use of a common test collection for all the systems in comparison. As the evaluation is affected by the changes in the test collection, we need an adequate changing test collection and evaluation methodology to interpret these results. In this project, we propose a method to create Evolving Test Collections and a Continuous Evaluation Framework to analyze the performance difference of systems evaluated in different test collections.

2.6 Conclusion

In this Chapter, we have presented the state of the art in the Evaluation of IR Systems. The performance evaluation relies on different metrics, and different methods were proposed to define if a system S1 outperforms a system S2. Classical evaluation limits a continuous evaluation because it requires evaluating all the compared systems using the same test collection, as the performance of the systems is affected by changes in the test collection 2.3. As presented in the Chapter 1, this research address this problem:

> We can not compare the performance of a system evaluated in one test collection to a system evaluated in a second test collection because the performance variations are dependent on the changes in such collection. There is no evaluation framework to compare systems evaluated in evolving test collections that also explains how the changes in the evaluation environment affect the performance results

To our knowledge, there is no evaluation framework that continuously interprets the performance of a system evaluated in different collections. Standardization 2.4.2 tackles the problem of comparing queries of different difficulty, but it does not show how to interpret the effect of different test collections in the evaluation of different systems. Meta-analysis and Reproducibility metrics 2.4.3 are focused on extracting an overall effect of the test collections, but they require evaluating all the compared systems at each test collection. This setup is different from our continuous evaluation, which is tackling the problem of a Web continuous evaluation where the systems and the test environments change.

According to this chapter revision, the literature does not provide suitable test collections to perform a continuous evaluation of systems (in section 2.5). Which is related to another problem defined in Chapter 1:

> There is no test collection with controlled evolutions of the set of documents, queries, or relevance judgments available in the literature in which we can apply and test a continuous evaluation of information retrieval systems.

We realize that there are static test collections that could be adapted to simulate the dynamic Web environment. Test collection maintenance, in section 2.4.1, proposes a method to update a current collection incorporating dynamicity to the test collection. We build upon these works to create controlled evolving test collections that can be used to evaluate and compare the performance of systems continuously.

Chapter 3

Evolving Test Collections

This thesis aims to study the evaluation of IR systems on collections that evolve over time. In order to continuously evaluate a system, we will rely on an Evolving Test Collection (ETC), in a way to analyze and to compare the system's performance in evolving contexts. This chapter focuses on defining and building such ETC. We present the use case that sustains the construction of these test collections in section 3.1. We formally present the elements that define an ETC in section 3.2. Section 3.3 describes how to proceed to simulate ETCs from existing static test collections. Using the formalization defined, section 3.4 depicts the building of an actual ETC using an industrial seach engine (Qwant) data, the CLEF LongEval ETC[1], in which documents, queries, and judgments change according to the real changes.

3.1 Introduction

The continuous evaluation of a Web search engine is a challenging task because of the constant evolution of the submitted queries, documents, and relevance judgments: in 2017 Google published that 15 percent of the submitted queries in its system every day are new[2]; sets of documents are continuously added to the corpus of indexed Web documents. Moreover, documents are continuously being removed from the index, while others are modified [53, 81]. For instance, experiments by Ntoulas et al. [81] show that over one year, 20% of the Web pages from 154 popular Web sites are removed, and overall 50% of the content of the pages get modified.

As described in Chapter 2, to evaluate an IRS, the Cranfield paradigm is followed most of the time: such classical offline evaluation method is based on test collections composed of a corpora of documents, a set of queries and a set of relevant judgments that define which documents are relevant for the tested queries. This framework corresponds

[1] https://clef-longeval.github.io/

[2] https://blog.google/products/search/our-latest-quality-improvements-search/

to a static evaluation effort that provides stable and reproducible results. A static test collection represents an independent evaluation attempt in which all the evaluated systems are compared.

The first problem described in a continuous evaluation (Chapter 1) is related to the evolution of a real Web search scenario. As described in section 1.2, such environment is dynamic and the hypothesis of using a static test collection is not valid, as changing environments impact the performances [45, 133] (section 2.3). In such a case, repeating the evaluation is needed to update the performance of the search engine according to the changes in the environment. Then, the system's performance is described as a list of values in time that changes according to snapshots of the test collections. The work reported here explores offline evaluations on evolving test collections in a similar way to classical Cranfield evaluations in order to see how the evolution of the parameters impacts the performances of the evaluations of systems.

Web search engines could be continuously evaluated using the offline paradigm using snapshots of the evolving collection: it is then like building different and independent evaluation environments. In terms of the stability of the performance of a system, it is usually evaluated using *shards* (described in section 2.4). The shards creation do not control the differences between the created test collections, and then, the evolution of their components is not exploited. Rashidi [88] showed that feature-based test collections present more different results than random sub-collection, but still, they propose to split one collection. In this chapter, we aim to build a test collection to answer a fundamental question on the *robustness* and *stability* of Web IR systems against the evolution of the data.

RQ How does performance of an IRS behave as the collection evolves? Such a question is especially important for commercial systems, as users' satisfaction is the core of such systems.

We propose a way to take into account the evolution between test collections, according to a few parameters, and to study their impact on the stability of systems performance. As we know that the differences among the test collections change the performance of the system, we propose to fix these differences between test collections with an evolving criterion in order to analyze if we are able to get more information about the performance of the system when evaluating a system in an ETC. As no such previous work exists, to the best of our knowledge, we need to study the evolution in a controlled way to assess its interest. To do so, we focus on the simulation of ETCs to build our experiments in a controlled and reproducible way. Later on, we also describe the creation of a real ETC with the Qwant Search Engine to evaluate the evaluation framework in a real environment.

3.2 Formalization

In this section we formally define the concept of Dynamic test collection (DTC) and Evolving Test Collection (ETC).

Dynamic test collection: The concept of Dynamic Test Collection (DTC) was first defined by Carterette et al. [25]: a test collection is called dynamic if it simulates user interactions in response to the evaluated system. A different definition of dynamic is presented in the maintained test collections described in section 2.4.1. In such a case, a test collection is called dynamic if it has changing components across time. Maintaining a test collection does not rely on simulations by on updating the corpus of documents or relevance assessments with respect to real changes in the Web, after the creation of the Test Collection. In contrast with Carterette et al. [25], we aim to simulate changes in any component of the test collection by relying on all the components of existing test collections. Compared to maintaining a test collection, as the work of Soboroff [115], Hashemi [59] and Tonon [121], our point is not to update the test collection post its creation, but to extract dynamic features from the components that are already defined in the test collections.

We define a dynamic test collection as a list of several test collections that have variations between each other with the objective of evaluating a system in a changing environment. Therefore, a DTC extends the static test collection by defining a sequence of *epochs* representing each TC, varying the components in previous and following ones in the DTC. Following the definition of TC in section 2.2.1:

A **Dynamic Test Collection** (DTC) is a list of test collections (TC), where each test collection TC_i has three components (sets): documents (D_i), queries (Q_i) and relevance judgments ($Qrel_i$).

$$
\begin{aligned}
DTC &= (TC_1, ..., TC_i, ..., TC_n)_{1 \leq i \leq n} \\
&\text{where,} \\
TC_i &= (D_i, Q_i, Qrel_i), \\
D_i &= \{d_{i,j}\}_{1 \leq j \leq nd_i}, \\
Q_i &= \{q_{i,k}\}_{1 \leq k \leq nq_i}, \\
Qrel_i &= \{(d_{i,j}, q_{i,k}, f_i(d_{i,j}, q_{i,k}))\}
\end{aligned}
\tag{3.1}
$$

where,

$$
\begin{aligned}
f_i &: D_i \times Q_i \rightarrow AV_i \\
n &\in \mathbf{N};\ nd_i, nq_i \in \mathbf{N}, \forall i \in [1, n]
\end{aligned}
$$

with AV_i, the domain that lists the set of assessment values in TC_i.

Evolving test collection: We are interested in defining a specific dynamic test collection: an Evolving Test Collection which aims at supporting the evaluation of IR when the evaluation, and therefore, the test collection evolves. At each epoch, the documents, queries and relevance judgements change in an evolving and consistent way. In an evolving test collection, we expect features that can be computed in the components to measure the evolution of the test collection, for example, a set of overlapping elements from one epoch to the next one might define the change. Another important consistency factor is the existence of a unique set of assessment values (AV) across the epochs, formally:

An **Evolving Test Collection** (ETC) is a DTC with evolving documents, queries, and relevance judgments using a common set of assessment values AV.

Therefore, the AV values are shared for all the test collections in the ETC:

$$AV_i = AV, \forall i \in [1, n] \tag{3.2}$$

It is worth mentioning that, using the proposed definition, a classical Cranfield static test collection with binary assessments corresponds to an ETC with one epoch, leading to:

$$ETC_{\text{static}} = (TC), AV = \{0, 1\}$$

ETC Components: In the following, to keep the generality of our proposal, we will use the meta-notation C_i to denote any component D_i, Q_i or $Qrel_i$. As we are focusing on the evolution of test collections, we may describe TC_i, with $i \in [2, n]$, relative to its preceding epoch, namely TC_{i-1}. With TC_1 composed of initial sets of elements for each component C_1. Then, any component C_i of TC_i, is possibly described using the elements of C_{i-1} that:

- do not appear anymore in C_i, noted $C_{i-1,del}$;
- do appear in C_i but not in C_{i-1}, noted $C_{i,add}$;
- have their content modified between C_{i-1} and C_i, noted $C_{i,mod}$. An updated document or a modified assessment falls into such modifications.

More formally, C_i is defined as:

$$C_i = C_{i-1} \setminus C_{i-1,del} \cup C_{i,add} \tag{3.3}$$

Notice that we do not consider $C_{i,mod}$ in the definition of C_i, as the elements belong to C_i and C_{i+1}, but they are a different version of the element. Finally, we group the component list from consecutive epochs to define dynamic components (DC) for the documents, queries, and relevance judgments as:

$$DC = (C_1, ..., C_i, ...C_n), \text{with } C_i \in TC_i$$

The DC notation is used in the next section 3.3 to define how to simulate an ETC using a static test collection.

3.3 Simulation from existing TC

This section proposes a method to build an ETC from an existing static test collection. The purpose of creating an ETC is to study the changes in the systems' performance using our evaluation framework (defined in the next chapter 4). In section 3.3.1, we define the simulation procedure; in section 3.3.2, we depict the features of different types of simulation. Considering the general definition and the features, we propose two implementation methods in section 3.3.3 and three use cases that connect the simulation with the state-of-the-art in section 3.3.4. Finally, in section 3.5.1, we present some experiments using different ETC configurations to analyze the performance behavior of an IRS as the test collection evolves according to the ETC parameters.

3.3.1 Simulation procedure

In this section, we simulate an ETC in a controlled manner. In our case, a simulation builds an ETC according to a source static test collection TC^s and a given set of parameters that controls the ETC components' evolution across a number of n epochs. This controlled evolution allows us to study precisely the behavior of the evaluated systems.

Following the ETC definition, we create an ordered list of TC_i from TC^s on n evolving epochs. As defined in the state-of-the-art (section 2.2.1), a TC^s is composed of a set of documents (D^s), a set of queries (Q^s), and a set of relevance judgments ($Qrel^s$), we denote them with an "s" to indicate that they are part of the static test collection used as the source of the simulation. For example, Figure 3.1 presents a simulated ETC with three epochs ($n = 3$) from a TC^s.

At each epoch $i \in [1, n]$, a simulated TC_i is defined, which comprises a subset of elements for each source component (C^s). The components of the simulated TC can change from one epoch (TC_i) to the next one (TC_{i+1}). We control the evolution of the ETC with the vector $N_c :< nc_i >$ that defines the number of elements in a specific component at epoch i. Consequently, we simulate the evolution of each C_i as a list of subsets from C^s to create a dynamic component DC with the same length n. In practice, we simulate DD, DQ and $DQrel$, as dynamic components for documents, queries and relevance judgements, respectively:

1. $\forall D_i \in DD, D_i \subseteq D^s$ with $|D_i| = nd_i$

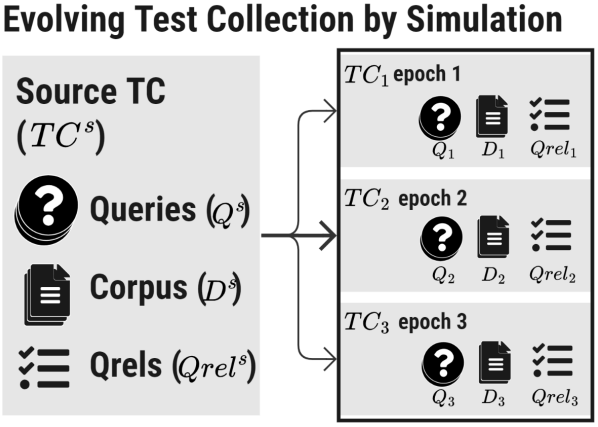


Figure 3.1: Creation of Evolving Test Collections by simulation.

2. $\forall Q_i \in DQ, Q_i \subseteq Q^s$ with $|Q_i| = nq_i$
3. $\forall Qrel_i \in DQrel, Qrel_i \subseteq D^s$ with $|D_i \times Q_i| = nd_i \times nq_i$

Creating an ETC by the variation of only one component is possible. In such a case, the other components remain constant across the DC to respect the creation of the same number of epochs (e.g., if only the documents change across epochs with a constant query set, then $Q_i = Q^s, 1 \leq i \leq n$).

A DC is built epoch by epoch, assigning a set of elements of the C^s into the component C_i. This construction is then dependent on the deleted elements from a past epoch ($C_{i-1,del}$) and the number of added elements in the current epoch ($C_{i,add}$). A specific strategy should define how to select which documents are added or deleted from the components to control the simulated ETC.

As defined above, any element of C_i belongs to its respective C^s. As a consequence, there is no modified document or query ($C_{i,\text{mod}} = \emptyset, \forall i \in [1, n]$). This is due to the fact that we rely on a stable TC, where the relevance is assessed for the specific $(q_{i,j}, d_{i,k})$ pair. This stable TC^s also provides a unique AV^s. Therefore we take the same set to define the assessment values of the simulated ETC. This construction is consistent with the definition of ETC as a DTC with a unique AV set, which is defined in equation 3.2.

Overall, we define the simulation $\mathcal{S}$ to create a simulated ETC, with the following parameters: a TC^s, a number of epochs (n), the size of each C_i (N_C), and a simulation *strategy* (described in section 3.3.2). Therefore, $\mathcal{S}$ is defined as:

$$\mathcal{S} : (TC^s, n, N_C, strategy) \rightarrow ETC$$

then a simulated ETC is described as:

$$ETC_{\text{simulated}} = \mathcal{S}(TC^s, n, N_C, strategy), \text{with } AV = AV^s$$

3.3.2 Strategy: Features and Constraints

We define the simulation strategy according to a set of constraints that defines the TCs that compose the $ETC_{\text{simulated}}$. These constraints may be used to analyze which changing component affects the system's performance. A strategy depends on constraints related to the *cardinality*, *overlap* and *ordering function*:

- Forcing the **cardinality** of some components to be equal, as in:

$$\forall nc_i \in N_C, \forall i \in [2, n], nc_{i-1} = nc_i \Rightarrow |C_{i-1}| = |C_i| \tag{3.4}$$

  In such case, we have that $|C_{i,del}| = |C_{i,add}|$. This is typically what may be defined if we want to study TCs that contain the same number of elements to avoid potential biases due to differences in the components' size.

- Considering one component, a constraint may fix a global **overlap** value o between successive components using:

$$\forall i \in [2, n-1], \frac{|C_{i-1} \cap C_i|}{|C_i|} = \frac{|C_i \cap C_{i+1}|}{|C_{i+1}|} = o \tag{3.5}$$

  This may be useful when comparing the results of successive TCs, as we know that they differ by the same number of elements. The overlap value o defines a controlled change between C_i and C_{i+1}. The number of overlapped elements between C_i and C_j decreases while $j - i$ increases.

- Depending on the collection, a source component C^s of TC^s may be described by additional features. For instance, an **ordering function** $\mathcal{F}_C$ may exist on one C^s:

$$\begin{aligned} \mathcal{F}_C : C^s &\rightarrow \mathbb{R}^m \\ \text{with, } m &= |C^s| \end{aligned} \tag{3.6}$$

  The function $\mathcal{F}_C$ gives to any element of the component C^s a (real) value that allows ordering these elements. The ordering is *strict* when F_C maps each element to a unique value in $\mathbb{R}$. Otherwise, if the ordering is *partial*, several elements are mapped to the same value. A typical example is a timestamp on the document's creation date. The ordering is *strict* if each timestamp is unique, and *partial* if a timestamp is repeated.

  Using such a function, we are able to know, for any subset of C^s, that forms a C_i in TC_i, the minimal and maximal $\mathcal{F}_C$ of its elements. Then, we may constraint any

component C of the ETC, stating for instance that all the elements of one set C_{i-1} have a lower $\mathcal{F}_C$ value that any of the elements of C_i:

$$\forall i \in [2, n], max(\mathcal{F}_C(C_{i-1})) < min(\mathcal{F}_C(C_i)) \tag{3.7}$$

To ensure such condition, no element from $C_{i,add}$ belongs to any TC_i, previous to TC_j with $i < j$, and the removed elements $C_{i-1,rem}$ are those with lowest $\mathcal{F}_C$ value in C_{i-1}. For instance, the condition (3.7) may be used to guarantee that TCs are containing successive timestamped documents.

All these constraints define the *strategy* that controls the simulation process $\mathcal{S}$ when creating an $ETC_{\text{simulated}}$. For instance, if we want to simulate the evolution of a Web corpus with documents that are created at one time, we may define that the *strategy* uses an ordering function $\mathcal{F}_C$ that gives the creation timestamp of each of the document's in D^s component, and we control the amount of overlap between successive TCs.

An $ETC_{\text{simulated}}$ may then be used to continuously assess the impact of the component's changes on the performance of the evaluated system. The interest of such a study is a) we may find out that the defined parameters of the ETC impacts the quality of systems and, more importantly for us, the variability of the evaluations, and b) the classical usage of generating sub-test collections may not be adequate to mimic the reality.

3.3.3 Instanciation

We propose to instanciate the simulation $\mathcal{S}$ to create an $ETC_{\text{simulated}}$, which is described with the following parameters:

$$\mathcal{S}(TC^s, n, N_C, strategy)$$

It is implemented in two ways, depending if the *strategy* constrains the overlap of elements or not. In the simplest case, the *strategy* only constrains the number of elements in each test collection of the ETC; we call it *Random ETC* (in section 3.3.3.1). The *strategy* that controls the number of overlapped elements uses the $\mathcal{F}_C$ function to order the elements of C^s; such ETC is called *Overlapped ETC* (in section 3.3.3.2).

3.3.3.1 Random ETC

A Random ETC mimics a changing environment, for example a set of documents that could be removed and re-incorporated into the test collection later, or queries that are season-related. This evolution is simulated by randomly extracting several samples from the TC^s test collection.

To create an Random ETC, we implement Algorithm 1 following the simulation *strategy* as:

$$strategy = (cardinality)$$

this *strategy* is only based on the *cardinality* that constraints the number of elements at each C defined in N_C. The algorithm 1 receives C^s as the source components, the number of epochs n as the number of subsets to create from C^s, the size of the component at each epoch as N_C, and finally, we incorporate the *sampling* method as a variable, which defines how to sample the elements from the respective component. The algorithm returns a dynamic component DC as a list of subsets from C^s.

Algorithm 1: Random ETC Simulation process for a component C from a source component C^s.

Data: C^s, n, N_C, *sampling*
Result: $DC = [C_1, ..., C_n]$
$C_1 \leftarrow$ a *sampling* sample of size nc_1 from C^s;
add C_1 to DC;
for $i \in 2..n$ **do**
  $N_{del} \leftarrow sampling(nc_{i-1})$;
  $C_{i-1,del} \leftarrow$ a *sampling* sample of size N_{del} from C_{i-1};
  $N_{add} \leftarrow nc_i - |C_{i-1,del}|$;
  $C_{i,add} \leftarrow$ a *sampling* sample of size N_{add} from C^s;
  $C_i \leftarrow C_{i-1} \setminus C_{i-1,del} \cup C_{i,add}$;
  add C_i to DC;
end

A generated Random ETC simulated by $\mathcal{S}$ creates a first C_1 by selecting a sample of size nc_1 from C^s. For the component C_i of the next epoch, we select a sample $C_{i-1,del}$ of size N_{del} from the component C_{i-1} at the previous epoch, and a sample $C_{i,add}$ of N_{add} elements from C^s; then C_i is composed of the previous component set (C_{i-1}) minus the elements to remove ($C_{i-1,del}$) and adding the elements to add ($C_{i,add}$). This process is repeated a defined n number of epochs.

In this case, the *strategy* constrains the number of elements at each component and the *sampling* method is used to select the elements that belong to each component epoch. A special case occurs if having a constant component size, the *sampling* method defines that the number of elements to remove is equal to the component size ($N_{del} = N_{add} = nc_i$) then, the Random ETC is build as a random sampling.

A Random ETC does not control the actual number of overlapping elements because they can be sampled several times for different subsets of the component.

3.3.3.2 Overlapping (Ov.) ETC

When evaluating systems using a Random ETC, as we do not control the amount of overlap between consecutive TCs, it is difficult to assess the behavior of a system without knowing how similar the test collections are. Therefore, the random ETC simulates changing contexts but not evolution. An Overlapping (Ov.) ETCs simulates the evolution of the test collection by controlling the similarity between the component epochs as the overlap in the ETC: any two consecutive TCs pairs in the ETC are equally similar with respect to a similarity feature that needs to be defined.

To create an overlapping (Ov.) ETC, we implement Algorithm 2 following the simulation *strategy* as:

$$strategy = (cardinality, overlap, \mathcal{F}_C)$$

the *strategy* for creating an Ov. ETC relies on a full ordering of the component's elements $\mathcal{F}_C$, assigning a specific value to each element. It does not allow the reinsertion of elements from previous versions and defines a fixed number of overlapped elements across components epochs (*overlap*). As in random ETC, the *cardinality* is also constrained, N_C is a constant value.

Algorithm 2: Overlapping ETC Simulation process for a component C from a source component C^s

Data: C^s, n, N_C, $overlap$, $\mathcal{F}_C$
Result: $DC = [C_1, ..., C_n]$
$C^s \leftarrow$ order C^s with $\mathcal{F}_C(C^s)$;
$init \leftarrow 0$;
$end \leftarrow nc_1$;
$C_1 \leftarrow \{e_i | e_i \in C^s, init > i > end\}$, elements $init$ to end from C^s;
add C_1 to DC;
for $i \in 2..n$ **do**
  $N_{add} \leftarrow nc_i * (1 - overlap)/100$;
  $N_{del} \leftarrow nc_{i-1} - nc_{i-1} + N_{add}$;
  $C_{i-1,del} \leftarrow \{e_i | e_i \in C_{i-1}, init > i > N_{del}\}$, first N_{del} elements from C_{i-1};
  $C_{i,add} \leftarrow \{e_i | e_i \in C^s, end > i > N_{add}\}$, next N_{add} elements from C^s;
  $C_i \leftarrow C_{i-1} \setminus C_{i-1,del} \cup C_{i,add}$;
  add C_i to DC;
  $init \leftarrow init + N_{del}$;
  $end \leftarrow end + N_{add}$;
end

First, the elements of C^s are ordered by a specific feature defined by $\mathcal{F}_C$, then a first C_1 is created by selecting the first nc_1 elements from C^s. For the next component i (C_i),

we compute the number of elements to add (N_{add}) and to remove (N_{del}) to respect the constrained *overlap* between C_i and C_{i-1}. N_{add} and N_{del} are computed using the *overlap* and the size of the current (nc_i) and the previous component (nc_{i-1}). We select the first N_{del} elements from the previous component set (C_{i-1}) as $C_{i-1,del}$, and the next N_{add} elements from C^s as $C_{i,add}$; then C_i is composed of the previous component set (C_{i-1}) minus the elements to remove ($C_{i-1,del}$) and adding the elements to add ($C_{i,add}$). This process is repeated for each of the n epochs.

In the proposed implementation of an Ov. ETC simulation, we constrain the ordering of the component, the cardinality, and the number of overlapped elements from successive components. A special case occurs when the overlap is 100%. In this case, if the cardinality grows, then the simulation exemplifies a growing evolution of the component without removing elements ($N_{del} = 0$).

The proposed implementations depends on the *strategy* used on the simulation $\mathcal{S}$. Other implementations can be proposed considering other strategies, such as the use of repeated elements in the components. We only consider stratifying the source test collection.

3.3.4 Simulation Cases

The implementations above create different $ETC_{\text{simulated}}$ by the definition of specific parameters in the simulation. As a way to exemplify different strategies, we present four examples of simulated ETCs. The first two examples are based on state-of-the-art sub-collections: sharding and bootstrapping (described in section 2.3.1). As a third example, we present a sharding Ov. ETC, this Ov. ETC is created to control the amount of change in the successive epochs that are defined by an overlap value. And as the forth and final example, we present a Ov. time-based ETC defined to control the changes between epochs with a temporal base as a mimic of Web dynamic environment.

Table 3.1 presents the four simulation examples: *sharding* Random ETC, *bootstrap* Random ETC, Ov. *sharding* ETC, and finally, Ov. *time-based* ETC, with their respective parameters. The table shows, for each example, the ETC implementation followed: Random or Ov. ETC; the parameters: *overlap*, and the nature of $\mathcal{F}_C$. The parameters n, and *cardinality* are not included in the table. n can take any natural number, which does not change the definition of the ETC. The *cardinality* constrain defines the N_C values as constant for all the simulation cases, therefore $\forall i \in [2, n], nc_{i-1} = nc_i$. We add the column nc_i, as the number of elements in each component with relation to the number of elements in C^s, and the column *duplicates* in the case that the ETC selects in one epoch any elements more than once for any component.

Sharding and **Bootstrap** are state-of-the-art strategies to create sub-collections. They are used to analyse the stability of the system's performance in the context of changes in the document or query sets of the collection. Using our simulation process, we can simulate these sub-collections as Random ETC. As described in Table 3.1 rows 1 and

	name	*ETC*	*overlap*	$\mathcal{F}_C$	nc_i	*duplicates*
1	sharding	random	-	-	$\|C^s\|/2$	no
2	bootstrap	random	-	-	$\|C^s\|$	yes
3	overlap sharding	overlapping	constant	random	$\|C^s\|/a$	no
4	time-based	overlapping	constant	time	$\|C^s\|/a$	no

Table 3.1: Random ETC and Ov. ETC simulation examples and their parameters, with $a < |C^s|, a \in \mathbb{N}$.

2, these Random ETCs do not control the overlap of the components, the size of the test collection components is the same across the epochs, and there is no order to select the documents at each epoch. The only difference between *sharding* and *bootstrapping* is that in the first one, there is no repetition of elements (*duplicates*), while in bootstrapping, it is allowed to duplicate elements (as it is defined in the bootstrapping process). This duplication of elements makes possible to create several samples from C^s with the same size ($|C^s|$).

In the third row of table 3.1, we present the parameters of an sharding Ov. ETC. This Ov. ETC is created to control the amount of change in the successive epochs that are defined by an *overlap* value. We take inspiration from *sharding* and *bootstrapping* to define more meaningful ETCs by controlling small changes from one to the next test collection. Still, we do not know the relation between the elements in each component, because the creation of the ETC follows a random creation. The motivation to create this specific sharding Ov. ETC is related to the need to control the overlap between test collection epochs and evaluate how a controlled addition or removal of elements in one component affects the performance of systems. This control is not possible with *sharding* and *bootstrapping*. This is an Ov. *random* ETC that works as a baseline to evaluate the Ov. *time-based* ETC.

In the final row of table 3.1, we present a Ov. *time-based* ETC defined with an ordering function for the components elements by a time-based feature to mimic the real Web dynamic environment. We focus on simulating an ETC that approximates the evolution of the Web in a controlled manner. The simulation of the Ov. *time-based* ETC defines the same parameters of a sharding Ov. ETC, but at least one component uses a function $\mathcal{F}_C$ to map each element with a timeline. Ov. *time-based* ETC controls the *overlap* between successive component epochs and maps a date for each element to allow the ordering of the component (e.g., the creation date of a document defined in the source test collection). We are then able to study an ETC that is closer to the real case of the Web.

To conclude, a Random ETC simulation helps us to create state-of-the-art ETCs that are used to analyse the stability of the test collection, but without control of the changes.

To explore different cases, we propose an Ov. ETC based on an ordering feature and an *overlap* value that controls the amount of changing elements. *Sharding* Ov. ETC is presented as a general ETC that controls the number of overlapped elements across the ETC. Finally, we present Ov. *time-based* ETC as a closer simulation of the Web environment.

3.4 ETC Acquisition: CLEF LongEval

In this section, we present LongEval [50], a new Evolving Test Collection with data acquired from Qwant Search engine. The LongEval ETC is proposed to support the evaluation of commercial and open-source state-of-the-art Web IR systems. This ETC is dedicated to provide a large-scale evaluation and is able to cope with the temporal evolution of real Web data. Several organizations contributed on the construction of LongEval. The main contribution of this thesis is the design of the acquisition process and the definition of the topics that lead the gathering of documents, queries and relevance judgements across time.

LongEval follows the definition presented in section 3.2. It is built as a succession of three test collections, each of them composed of a set of documents from Qwant's actual index, a set of topics acquired from actual user's queries, and two sources of relevance judgements. It is composed of a large amount of collected, filtered and cleaned Web pages (several millions) in two languages (original French documents and their automatic English translations). The high-quality translations of queries and documents might be helpful for researchers working on cross-language retrieval. The ETC is designed to reflect the changes of the Web across time, by providing evolving documents, query and relevance judgement sets.

Such a large ETC collection (millions of documents, thousands of queries, large relevance judgement sets) is usable as a good training source for deep learning IR methods. Though our main focus is Web search, performance evaluation using LongEval ETC can help to study in detail the robustness of IR models against novelty (documents, queries). Such data is necessary for the community as this question is still largely open for IR.

As discussed in section 2.5, and to our best knowledge, the only collection with similar temporal settings is the recent TREC-COVID dataset [127]. Compared to the LongEval collection, the number of documents and queries in TREC-COVID is rather small (few tens of thousands of documents and 50 topics) and focused on studying a very specific topic, the COVID outbreak. In our case, the queries are much broader and based on a commercial search engine query logs. Thus, LongEval ETC is the only large collection with up-to-date data (acquired in 2022) that exists to evaluate modern IR retrieval on evolving data.

3.4.1 Data Acquisition Overview

In this section, we describe the general acquisition process of the data from the Qwant Web search engine and the creation of different collection components. The overall acquisition is periodic and is recurrent over time to build each test collection that composes the ETC. The queries, documents and relevance judgments change across the test collections, but they are acquired with respect to their relation to a defined set of topics (which are a keyword that describes a broad and general concept).

3.4.1.1 Data description

The data overview is presented in Figure 3.2. It consists of:

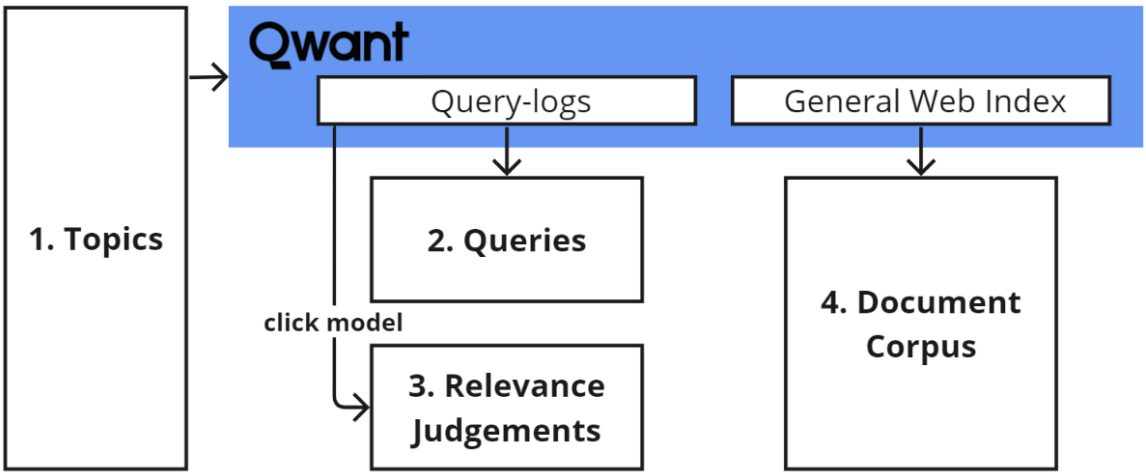


Figure 3.2: Data acquisition general strategy. Qwant private resources are in blue. White boxes represent the test collection elements.

1. The acquisition of a set of **topics**, selected from the Web and social media. This acquisition is based on trending – yet stable in the long-term – topics and is performed only once for the entire LongEval collection. The selection of the set of topics is further described in Section 3.4.2.1;

2. The selection of search **queries**, related to the topics, coming from the actual queries issued by Qwant users. We detail the query selection method in Section 3.4.2.2;

3. The creation of **relevance judgements**. We rely here on two approaches: implicitly using Click Models [33] computed from Qwant query-logs, and explicitly using manual judgements, which are conducted after the submission of this manuscript. Since each test collection contains several thousands of queries, explicit assessments

will be performed on a limited subset of manually selected queries. We present our methodology for generating relevance estimates in Section 3.4.3;

4. The acquisition of the **document corpus** (Section 3.4.4). This corpus is a union of: i) all the Web documents that have been displayed for each query of a test collection, and ii) a sizable random sample of the Qwant index. This protocol leads to a corpus that contains a mixture of relevant and non-relevant documents. The presented process handles the evolution of the Web pages, as the corpus is not only composed of URLs but also of the Web page contents acquired at a specific time.

As described, LongEval is built to evaluate systems along time. To do that, the acquisition presented is achieved periodically, typically each month. In each time period t, we create a collection composed of the queries, relevance judgments and documents collected during this month. The full LongEval ETC, composed of a sequence of collections, is thus dynamically evolving. This allows us to create and provide test collections for different time periods. The temporal acquisition is presented in Figure 3.3, LongEval is acquired in three periods, a time t, a second time t' as a short term acquisition and a third acquisition period as t" as an acquisition in a long term, further details are presented in section 3.4.6.

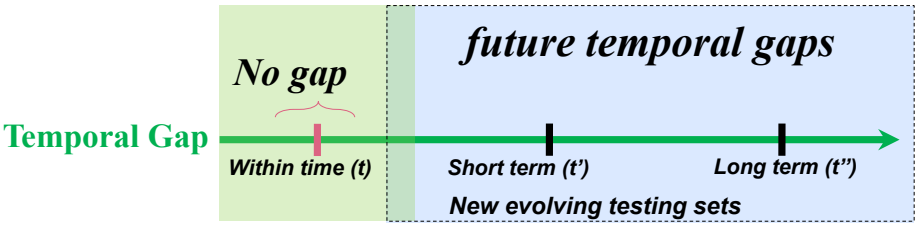


Figure 3.3: Global temporal acquisition framework for LongEval [4].

3.4.1.2 Acquisition description

Three teams participated in the creation and distribution of LongEval DTC: UGA (Université Grenoble Alpes), Qwant and LINDAT/CLARIAH-CZ[3]. This thesis is part of the UGA team, Qwant is the search engine to acquire the data, and LINDAT/CLARIAH-CZ is the Digital Research Infrastructure for Language Technologies, Arts and Humanities from Charles University. My main contribution to this task is the design of the acquisition framework and the definition of Topics.

The acquisition framework, including the main responsibilities of each team, is described in figure 3.4. In detail, the process is:

[3] https://lindat.mff.cuni.cz/

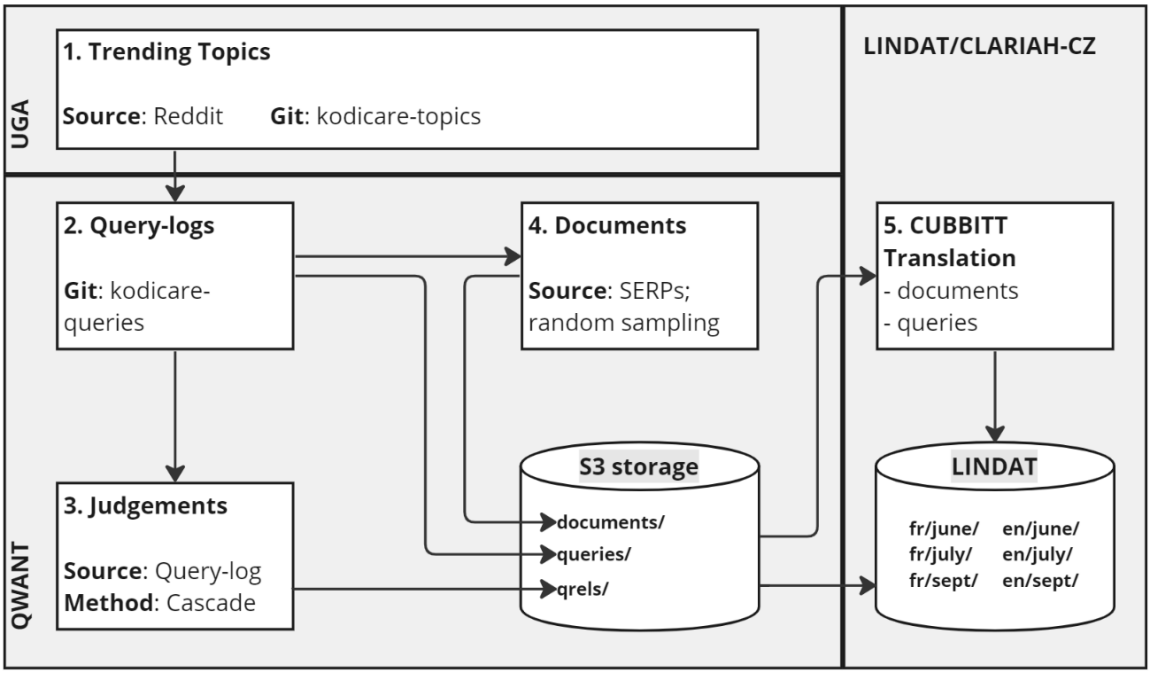


Figure 3.4: Acquisition framework and interaction between UGA, Qwant and LINDAT/CLARIAH-CZ.

1. The process starts with the acquisition of **trending topics** from social media by the UGA team, who is responsible for proposing a list of popular, general and stable topics for continuous evaluation;

2. Qwant searches in its **query-logs** for queries related to the trending topics. The selected **queries** are stored in a shared server.

3. The extracted query-logs are used to compute a set of **relevance judgements** using cascade click models. The result of the click model is shared in the storage server.

4. The extracted query-logs are used to sample the **document collection**, plus the negative sample from the index. All documents are in the shared server.

5. Finally, LINDAT/CLARIAH-CZ performs the translation of French documents and queries into an English version using CUBBIT system[4].

The LongEval ETC is finally stored in LINDAT server[5]. The ETC is freely accessible,

[4] https://lindat.mff.cuni.cz/services/translation/
[5] https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-5010

with data gathered in June, July and September 2022. We provide French and English versions. In the following sections, we describe the acquisition of each part of LongEval.

3.4.2 From Topics to Queries

As described above, the first step of the data acquisition is defining a set of *topics*, which act as a proxy for controlling the themes expressed in the queries. Those topics are later used for filtering and selecting the Qwant actual user's *queries*. The Qwant search engine processes approximately 200,000 queries per day. These queries follow the conventional long-tail distribution. Therefore, we need to define a way to sample interesting and challenging queries for which we would likely observe multiple user interactions. We decide to select the (query) topics corresponding to trending topics (i.e., topics that are popular among the queries asked to Web search engines) in English and French. This choice supports our use case of studying Web search engines, while it is also a good choice for examining temporal changes.

Definition 4 *In the context of the LongEval ETC, a topic is a short multi-word term – composed of either one or a few tokens – with a broad and potentially ambiguous meaning.*

The topics then serve as an entry point for selecting a different set of queries for each test collection. We allow the ETC to have a shifting query set from one collection to another to follow the potential drifts in the query distribution.

Definition 5 *In the context of the LongEval ETC, a query is a multi-word chain of characters that is related to one or more topics.*

We detail how we selected the topics of LongEval in Section 3.4.2.1, before presenting in Section 3.4.2.2 the method we used to define a set of queries for each topic.

3.4.2.1 Topics Selection

The Topics are selected based on the balance between three elements: *popularity*, *stability*, and *generality*, which are computed using Qwant query-log and Web tools. The process is described, in general terms, as:

1. Selection of popular source for topic extraction in social media.
2. Extraction of popular keywords as trending topics.
3. Compute stability and generality for each topic.
4. and finally, validation by Qwant, through the existence of queries related to the topics in Qwant query-logs.

The selection is based on gathering popular topics:

Definition 6 ***Topics should be popular****: this criterion should ensure that there will be enough potential queries covering this topic in Qwant's query-log.*

To select popular topics, we use external tools that show trendy topics according to several Web platforms, like Google Trends[6], a real-time daily and weekly index of the volume of queries that users enter into Google [32]. We targeted long-term trends of diverse domains to avoid selecting topics related to sudden interest shifts so as not to bias our collection towards events. The side effect of choosing such topics is that we will have a high probability of finding several Qwant users searching queries related to the topics on various timestamps in Qwant search logs.

The set of topics is selected from English and French languages to ensure a multilingual validation of our findings, although a large majority of Qwant users are French. To ensure some degree of consistency in the entire ETC, we defined a single set of topics that is common to all epochs. The list of topics was generated once for the entire LongEval collection in May, 2022.

To access trending topics, Reddit is a natural choice to access trending topics and to explore the initial phase of any event [86]. Reddit is also commonly used to get current trending topics in the medical field [22, 28, 71], extracting keywords from specific subReddit forums.

To access Reddit we use Python language and specific open libraries created to access the API of the service:

- **Reddit**: Reddit API[7] is directly accessible with Python requests. Using this API, we access posts and comments from a specific subreddit forum. We extract keywords from the latest posts of the most popular subreddits, in terms of active users.

There is no definition of trending topics in Reddit, only popular posts and forums. Then, we define which of the extracted keywords represent a trend given their recurrence across time. We follow a similar setup to [22, 28, 71], and use the Reddit API to extract keywords from the latest posts of the most popular subreddit forums in terms of active users. We select 15 popular subreddit forums with a general focus as News, science and open questions (listed in Table 3.2).

Trending topics are generated from popular keywords in subreddit posts. From each subreddit forum, a list of the 1000 top posts is accessed using the *top* endpoint[8]; *top* lists all post of the subreddit published in last *t=year* time-space based on the total number of negative and positive votes per post in descending order (listing 3.1 presents a post of worldnews subreddit). The title and description (named selftext) is stored to extract

[6] https://trends.google.com
[7] https://www.reddit.com/dev/api
[8] *https://oauth.reddit.com/r/subreddit/top?t=year*

	Subreddit name	# of subscribers
1	AskReddit,	36,553,897
2	worldnews	29,249,149
3	askscience	28,838,351
4	news	24,977,658
5	explainlikeimfive	20,945,436
6	technology	12,473,250
7	interestingasfuck	10,017,195
8	politics	8,136,655
9	Damnthatsinteresting	5,461,952
10	AskMen	3,998,306
11	AskWomen	3,821,687
12	europe	3,333,579
13	NoStupidQuestions	2,812,409
14	unpopularopinion	2,715,498
15	france	969,916

Table 3.2: Subreddits used for the topic selection.

keywords from. We only consider NOUN and PROPN as keywords[9]. For each post we store a maximum number of keywords (N=10), they are sort by pageRank value which ranks the importance of each node in a graph [83]. To do so, we use the content extraction algorithm TextRank, where each word of the post is a node in a graph, then PageRank algorithm is applied to rank each keyword [73], and the most important keywords are selected as the trending topics.

Listing 3.1: /r/worldnews post example.

```
{'subreddit': 'worldnews',
 'title': 'Macron wins French presidential election',
 'selftext': '',
 'upvote_ratio': 0.83,
 'ups': 139584,
 'downs': 0,
 'score': 139584,
 'is_video': False,
 'name': 't3_ub0f7j',
 'created': '24-4-2022'
 }
```

[9]We use spaCy library for Part-of-speech tagging, in https://spacy.io/

Definition 7 ***Topics should be stable in time:*** *it is important to consider topics with a consistent number of generated queries over time, so that we can really assess longitudinally the behavior of systems.*

To consider stable topics, we measure the persistence and recurrence in time on historical trending topics using Google Trends[10]. Google Trends provides a time series index of the volume of queries users enter into Google in a given geographic area [32]. Google Trends gives the likelihood of a random user to search for a particular search term from a certain location at a certain time[11].

To access Google Trends, we use Python language and a specific open libraries created to access the API of the service:

- **Google**: Pytrends[12] is a Python library used to access the Google API. This tool allows us to export 20 Daily trending Google searches for a specific country. Additionally, 10 trending topics per year are available.

Using Google Trends, we ask for the *interest over time* (IOT) of keywords. Google Trends returns historical, indexes data indicating when the keyword was searched most in a scale from 0, as no interest and 100 as the moment of highest interest. IOT represents the search interest on a keyword relative to the highest interest in a timeframe, but does not convey absolute search volume. We fixed a timeframe interest of one year, translated into 54 weeks (W) of interest. Then, we compute the difference between a perfect interest value versus the real one as:

$$IOT(kw) = |W| * 100 - \sum_{w \in W} iot(kw, w)$$

kw: keyword

Definition 8 ***Topics should be general*** *enough to cover numerous queries.*

We sampled queries from a 1-month Qwant's query log, and kept the queries that overlapped with our candidate topics. We considered topics for which a large number of queries matched ($\geq$ 1,000 queries) as general enough.

The code for the extraction and selection of topics is available in UGA GitLab[13]. Table 3.3 presents the set of topics that we used to further select the queries of the sub-collections.

[10] https://trends.google.com
[11] https://support.google.com/trends/
[12] https://pypi.org/project/pytrends/
[13] https://gricad-gitlab.univ-grenoble-alpes.fr/gonzagab/kodicare-topics

	Topic	English Description	subreddit
1	eau	water	r/explainlikeimfive
2	nourriture	food	r/explainlikeimfive
3	espace	space	r/explainlikeimfive
4	voiture	car	r/explainlikeimfive
5	argent	money	r/explainlikeimfive
6	manifestation	protest	r/europe
7	virus	virus	r/askscience
8	terre	earth	r/askscience
9	énergie	energy	r/askscience
10	police	police	r/news

Table 3.3: Example of 10 French topics of the LongEval-Retrieval collection that were the basis for selecting the queries. The English description is informative and has not been further used to build the collection.

3.4.2.2 Queries Selection

The topics previously extracted following the methods described in the previous section are a first step towards the identification of real user queries that are issued on Qwant. Qwant answers a portion of user queries with their own technology while the rest is forwarded to a third-party search engine. In order to ensure that all displayed documents are contained in Qwant's index, we restricted the query distribution that we consider to the queries that Qwant actually answers.

We use a simple text processing technique to map general topics to queries that are answered by Qwant. Let $\mathcal{Q}$ be the set of all queries that are answered by Qwant, and $\mathcal{T}$ be the set of topics we defined in Section 3.4.2.1. For each topic $t \in \mathcal{T}$, we select all the queries $\mathcal{Q}_t$ from $\mathcal{Q}$ that contain t as a sub-string (denoted by $\subseteq_{str}$):

$$\mathcal{Q}_t = \{q | q \in \mathcal{Q}, t \subseteq_{str} q\}$$

Then, for the full set of topics, we have:

$$\mathcal{Q}_{\mathcal{T}} = \cup_{t \in \mathcal{T}} \mathcal{Q}_t$$

As this filtering can lead to several tens of thousands of queries per topic when considering several days/months of query logs, we applied a top-k selection for each $\mathcal{Q}_t$, noted $\mathcal{Q}_t^{top_k}$, using which we only keep the k most frequently asked queries on Qwant for each topic. Finally, $\mathcal{Q}_{\mathcal{T}}^{top_k}$ denotes union of $\mathcal{Q}_t^{top_k}$ over the topics.

The selected queries came from Qwant clicklog. Qwant does not track information such as multiple clicks, dwell times, or query reformulations. The clicklog contains information about the search engine and the user clicks on the result of a Search Engine Results Page (SERP), such as the query, the displayed documents, the user location, the time of the day (timestamp), the position of the click, and the user's device. There is no user identification or session related to each click. A click is identified by a timestamp, a device (which can be either a smartphone or a desktop computer), and a position corresponding to the rank of the clicked document.

For a given period of time, Qwant filtered the clicklog files to retain all the entries corresponding to our previously selected queries, and dumped the appropriate information. Listing 3.2 provides an example of such data.

Listing 3.2: Excerpt of Qwant User Clicks for LongEval.

```
{
  'query': 'arrete grippe aviaire',
  'displayed_urls': [
       'https://www.ladepeche.fr/2021/12/31/grippe-aviaire-600-000-
          volailles-ont-ete-abattues-en-un-mois-en-france-10021018.php',
       'http://www.ladepeche.fr/2021/11/16/lot-mesures-contre-la-grippe-
          aviaire-ces-eleveurs-refusent-de-confiner-leurs-volailles-9931
          083.php',
       'https://fr.m.wikipedia.org/wiki/
          Diagnostics_et_prophylaxie_de_la_grippe_aviaire',
       'https://www.francetvinfo.fr/sante/maladie/grippe-aviaire/grippe-
          aviaire-six-nouveaux-elevages-touches-dans-les-landes_4895595.
          html',
       'https://www.humanite.fr/mot-cle/grippe-aviaire',
       'https://actu.fr/societe/yvelines-grippe-aviaire-eleveurs-et-
          autorites-sanitaires-en-alerte_47569193.html',
       'https://www.lafranceagricole.fr/actualites/elevage/grippe-
          aviaire-le-conseil-detat-rejette-les-recours-contre-la-
          claustration-des-volailles-1,1,3353476678.html',
       'https://www.20minutes.fr/societe/3208307-20211231-grippe-aviaire
          -600000-volailles-abattues-mois',
       'https://m.youtube.com/watch?v=IiegYNWtTg4'
   ],
  'locale': 'fr_fr',
  'clicks': [
      {
      'timestamp': '2022-02-14T09:34:46.357Z',
      'device': 'smartphone',
      'position': 9,
      },
      ...
   ]
},
```

Following the initial query selection process from Qwant clicklog, the queries undergo automatic filtering to select those with at least 10 relevance assessments (as detailed in the following section)[14]. Subsequently, a manual check is performed by a human annotator to ensure query quality. During this process, queries with similar objectives, such as 'achat voiture' and 'acheter voiture', 'anti virus' and 'antivirus', or 'bareme impots' and 'barême impots', are merged. Queries referring to adult content are removed from the collection.

3.4.3 Relevance Judgements

The LongEval ETC relies on user clicks as implicit feedback to automatically infer the relevance of different documents. Nevertheless, Raw clicks cannot be used as a signal of relevance due to their tendency to be noisy and heavily biased [61**?**] towards the top-ranked results. Noise comes from the fact that a click does not necessarily indicate relevance, while a lack of click does not indicate irrelevance. Aggregating larger samples of query logs can remove noise, but statistical biases remain. Such biases can result from various factors, such as position bias towards top-ranked results, presentation bias towards visually appealing results, or trust bias towards results from familiar domains [82].

To tackle this problem, Qwant provides implicit relevance judgements estimated from the clicklog (as exemplified in Listing 3.2). The click data is *debias* estimating Click Models. A Click Model [34, 139] is the base to infer the user relevance of a document from search log data. It computes the estimates of a document attractiveness given a query and thus it well tackles the problem of being able to use the users' interaction while avoiding sharing private data and reducing noise and bias. Many click models have been developed over the years with the goal of better modelling the clicking behaviour of users, mostly by investigating sessions of multiple clicks for a given query. Since Qwant does not track any search session, we cannot rely on the more advanced Click Models that consider multi-query and multi-click sessions.

Qwant thus implemented the simplified version of the Dynamic Bayesian Networks (DBN) [30] with a session length of 1, which comes down to the original Cascade Model [40]. The Cascade Model assumes the following user model: users scan a Search Engine Results Page (SERP) from top to bottom, skipping non-attractive documents and clicking on what they believe will be relevant based on attractiveness.

Using the click model, Qwant provides two relevance estimations: probability-based and discrete, for all query-document pairs. We specifically used 2 degrees of relevance, which correspond to:

- 0 = not relevant,
- 1 = relevant,

[14]It should be noted that some queries may have fewer than 10 assessments in the final collection due to further corpus filtering applied after this step.

- 2 = highly relevant.

The Listing 3.3 details raw relevance judgements computed by Qwant. The `relevance` attribute corresponds to the α parameter of the Cascade Model while the `relevance_dcg` corresponds to its discrete version using our method. The `n_seen` attribute derives from the Cascade Model and quantifies how many times a document has been seen by a user whether it has been clicked or not. This allows us to further quantity our confidence in the relevance estimates computed from the clicklog.

Listing 3.3: Excerpt of Relevance Judgments.

```
{
  "query": "abri voiture",
  "locale": "fr_fr",
  "serp": [
    {
      "finalUrl": "https://www.abri-arcis.com/",
      "relevance": 0.3333333333333333,
      "relevance_dcg": 1,
      "n_seen": 3
    },
    {
      "finalUrl": "https://www.homify.fr/livres_idees/24280/quel-abri-
          choisir-pour-sa-voiture",
      "relevance": 0.3333333333333333,
      "relevance_dcg": 1,
      "n_seen": 9
    },
    {
      "finalUrl": "https://www.abridejardin.pro/Garages-et-Abris-
          voitures/Voir-tous-les-produits.html",
      "relevance": 0.3076923076923077,
      "relevance_dcg": 1,
      "n_seen": 13
    },
    {
      "finalUrl": "https://fr.wikipedia.org/wiki/Abri_de_jardin",
      "relevance": 0,
      "relevance_dcg": 0,
      "n_seen": 6
    },
    ...
}
```

The last step of building the relevance estimates is their post-processing according to the filtering done on the queries. As described above, the queries were manually checked, and some were removed. In such cases, all the relevance estimates corresponding to such queries were also removed. Similarly, some of the queries were merged, and in such cases,

the corresponding estimates were also merged. If, during the merge, the assessments of the queries were having contradictory values (e.g. the document was relevant according to one query and not relevant according to the second query), these relevance estimates were excluded from the collection.

3.4.4 Document Corpus

The final part of the test collection is related to the corpus of documents, on which the relevant documents are selected to be retrieved for a specific query.

The first step for creating the document collection is to extract from the index the content of all the documents that have been displayed in SERPs for the queries that we selected (see Section 3.4.2.2). In addition to these documents, potentially non-relevant documents are randomly sampled from Qwant index in order to better represent the nature of a Web test collection. To do so, for each topic $t \in \mathcal{T}$, a maximum of $n = 100,000$ documents were randomly selected among those that matched the word tokens of t[15]. Such selection avoids oversimplifying the corpus and the search task, as these documents are not completely randomly picked from Qwant index.

The collection does not only provide the URL of each document in the corpus but also cleaned versions of these documents. To do so, we first extract the text content from the websites, for which we use the internal Qwant implementations. Thus, we are able to use exactly the same representations of documents as Qwant uses for ranking the documents. Listing 3.4 presents an example of the content of a document in the LongEval corpus.

Listing 3.4: Example of the content of a document.

```
{
  "url": "https://www.myamericanshop.be/collections/chips/
     products/pringles-loud-fiery-chili-lime",
  "content": "PRINGLES LOUD FIERY CHILI LIME My American Shop
      Epuise 3,29e Contactez moi lorsque ce produit est
     disponible:  \"Ce n'est jamais une bonne idee de jouer
     avec le feu !\",voila son slogan. Si vous aimez le
     piquant et les piments ces chips sont faites pour vous!
     Produits similaires S'INSCRIRE A LA NEWSLETTER",
  "created_at": 1585889589,
  "last_updated_at": 1586272033
}
```

The document collection construction process applies filters to remove adult and spam content. Even though this filtering is quite strict, adult and spam content is still very

[15] Qwant uses a very basic AND matching, hence selected documents had to contain all the word tokens of the topic.

frequent in the collection.

3.4.5 English Translations

Given that the vast majority of Qwant users are French speakers, the search engine's primary focus is on searching for and analyzing French queries and data. However, to make the collection more accessible to non-French speakers, the LongEval DTC provides English translations for both queries and documents. We create an English counterpart of LongEval using a machine translation system to translate French queries and documents into English. The translation is performed by the French-English CUBBITT (Charles University Block-Backtranslation-Improved Transformer Translation) system [85], available at the LINDAT/CLARIAH-CZ infrastructure[16]. Naturally, the quality of the translations of the queries, which are often very short, is much lower than the quality of the document translation. This is due to the mismatch of domain of training data which are not suitable for query translation. In the case of the documents, the translation is better due to the contextual information of the documents. Experiments conducted in LongEval [50] demonstrate that this approach has little impact on the evaluation measures.

3.4.6 CLEF LongEval 2023 shared task

LongEvalCLEF [4] is a shared task organized in September 2023 at CLEF. The task is dedicated to evaluating the temporal persistence of Information Retrieval (IR) systems and Text Classifiers. LongEvalCLEF is an opportunity to evaluate the creation of our Evolving Test Collection. Here, several participating systems did send runs using LongEval test collections created over sequential time periods, which allows doing observations at different time stamps t, and most importantly, comparing the performance across different time stamps t and t'. Submitted systems are evaluated in two scenarios: *short-term persistence* and *long-term persistence*. The *short-term persistence* task aims to assess the performance difference between t and t' when t' occurs right after or shortly after t. In the *long-term persistence* task, we aim to examine the performance difference between two t and t'', when t'' occurs several months after t (and thus $|t'' - t| > |t' - t|$).

As described in the overview (section 3.4.1), the acquisition is performed at each timestamp t, with t as a single month. We repeat this data collection process over several months and create the train collection t ($t = June$) and two test collections t' ($t' = July$) and t'' ($t'' = September$). Figure 3.5 shows LongEval ETC, including the three test collections:

[16] https://lindat.cz/services/translation

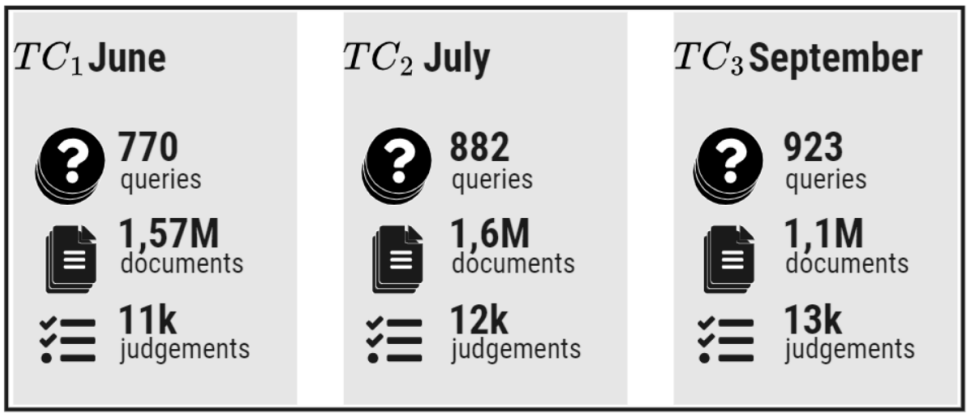


Figure 3.5: LongEval Evolving Test Collection.

June Collection (t). June collection is used as the train collection in CLEF LongEval. It was collected during June 2022 and released on the Lindat infrastructure[17]. The document corpus consists of 1,570,734 Web pages, 770 queries and 11,076 implicit relevance judgements. The queries are shared to the participants in two sets: 672 train queries and 98 heldout queries used to compare the performance of systems between t and t' or t''. 9,656 assessments correspond to the train and 1,420 assessments to the heldout queries. There are, on average, 14 assessments per query. About 73% of the assessments are non-relevant (8,050 assessments), 21% are relevant (2,354 assessments), and 6% are highly relevant (672 assessments).

July Collection (t'). July collection is used as to compute a short-term persistance in CLEF LongEval. It was collected over July 2022 and released on the Lindat infrastructure[18] as part of the LongEval test collections. The document corpus consists of 1,593,376 Web pages, 882 queries and 12,217 implicit relevance judgements. The relevance assessments are distributed in a similar way as June collection composed of 8,847 non-relevant, 2,608 relevant and 762 highly relevant assessments, with 14 average assessments per query.

September Collection (t''). September collection is used as to compute long-term persistence in CLEF LongEval. It was collected over September 2022 and released as part of the LongEval test collections on Lindat[2]. It is composed of 1,081,334 documents, 923 queries and 13,467 implicit relevance assessments (9,632 non-relevat, 2,899 relevant and 936 highly relevant).

[17] http://hdl.handle.net/11234/1-5010
[18] http://hdl.handle.net/11234/1-5139

Finally, we show the final structure of the LongEval ETC, built in the context of the CLEF LongEval shared task:

$$\text{LongEval} = (TC_{june}, TC_{july}, TC_{sept}),\ AV = \{0, 1, 2\}$$

3.4.7 LongEval ETC Limits

LongEval is an evolving test collection acquired based on trending topics and a commercial search engine (Qwant). We identify inherent limits related to the acquisition steps.

The trending topics are a set of keywords acquired from Redddit user posts. After a validation process, such topics are related to queries submitted into Qwant search engine by real users. Any change in the meaning of a keyword is a sensitive issue in our acquisition framework. We assume the *interest over time* is constant if a keyword does not change its meaning, at the same time that evaluating that a set of queries are still existing in Qwant click-log, even if the queries are different. In the case of change in the *interest over time* of a topic keyword, we need to delete the topic for the next test collections, and removing the documents, queries and assessments associated to it.

LongEval query sets are composed of real users of Qwant search engine. Even when the Qwant user are anonymised, some queries could contain sensitive information. In this case, we will remove the associated query and provide an update to the collection. In the same line, the lack of user sessions lead to the creation of implicit relevance judgments based on a simple click model. The method cluster clicks from different users, leading to possible noise and assessment errors.

Documents, queries and assessments are strongly related to Qwant resources. We expect to have reduced this bias by (i) extracting the trending topics using a first social network and validating the stability of a topic over a second web source defined as a broad general keywords, (ii) extracting thousand of related documents for each topics directly from the index and not from Qwant SERPs, and (iii) providing explicit relevance judgments as a counterpoint of the implicit relevance judgments.

3.5 Experiments

In this section, we present experiments to validate the ETC simulation process 3.5.1 and we present the evaluation results in LongEval ETC 3.5.2.

3.5.1 Simulated ETC Validation

In this section, we show the potential interest in considering ETCs for the continuous evaluation of the IRS. Our interest is focused on the evaluation of the stability of a system depending on the ETC configuration, comparing the state-of-the-art ETCs versus our

proposed temporal-based ETC simulation. It is of our interest to assess the relationship between the changes in the test collections and the performance of the system. Therefore, we explore the following research questions (RQ) using our simulation process:

RQ1. Are the temporal-based simulated ETC showing more different behavior of the evaluation of the systems than state-of-the-art test collections?

RQ2. Are the evolutions in a temporal-based simulated ETC related to a difference in the performance of the evaluated systems in the ETC?

For *RQ1*, we measure the stability evaluation as a way to understand the number of different performance values related to a system when it is evaluated in different collections. For *RQ2*, we measure the stability of the evaluation results contrasted to identifiable characteristics of the test collection epochs to understand the relationship between such features and the performance difference. The goal is to find a characterization for two different test collections that can imply significant changes across the evaluated systems (different performance values of ranking of systems). We find experimentally that the behavior between simulated evolving and independent epochs differs from several points of view. This study shows that we need to more precisely characterize the evolving test collections to address real Web search continuous evaluation.

3.5.1.1 Stability Evaluation

To study the contrast between several simulated ETCs, We compute the variation in the performances of a system evaluated in the ETC throughout a *pointwise stability*, and a lag-based pairwise stability. For one system S evaluated in the ETC of n epochs and M a performance metric, let $x_{S,M} = (x_1, ..x_i, ..., x_n)$ represents the performance of system S evaluated in each TC_i using M, where x_i is the mean value of all the queries at Q_i.

Pointwise performance stability Pointwise stability measures the variation in the performance of the system evaluated in the ETC. The performance of a system changes according to the differences among the TCs that compose an ETC. We measure pointwise stability based on the standard deviation the perforamance of a system (S), as $\sigma(x_{S,M})$. $\sigma(x_{S,M})$ quatifies the variability of the performance evaluation scores of a system using an ETC. Using this descriptor of the performance variation, a large $\sigma(x_{S,M})$ value represents very different performance results across the ETC. In a stability evaluation, we try to find the ETC configuration that provides the larger number of different performances associated with an IRS.

We evaluate several ETC configurations and find that the Ov. time-based ETC provides more results than random ETCs. With Ov. ETC, we can control the difference between test collections in terms of the number of common elements. Then we measure

the "lag-overlap" effect. This is a "baseline" effect of overlapping elements. We hypothesize that any evolving change will be higher than the random at any moment of the collection.

Pairwise performance stability First, we check how the stability change when the test collections are more different for one system. We propose two pairwise metrics to evaluate the difference between the performance of a system evaluated in two TCs with relation to the distance of between the collections: a lag-based differential stability and a lag-based α-stability.

The previous metric describes the set of performance evaluations in general without considering the ETC as a list of n TCs. We propose to characterize how the performance is changing in the sequence of TCs, measuring the differences in performance across pairs of TCs. We evaluate if the performance change is related to the ETC by comparing performance in TC pairs of different distances. We hypothesize that more different TCs should provide an unstable change in the performance of systems. Proving that an evolution of the test collection is related to the evolution of the performance of the systems.

We measure the differential performance of a system as the relative difference in the performance of a system evaluated in TC_i versus TC_{i+lag}, with $i \in [1, i - lag]$:

$$\text{diff}_M(S, TC_i, lag) = (x_i - x_{i+lag})/x_{i+lag} \tag{3.8}$$

We consider pairs with different temporal distances, represented by *lag*. The lag defines the distance between two TCs, then we can measure how the differential value changes when the compared TCs are more separated from each other.

Finally, the standard deviation of $\text{diff}(S, TC_i, lag)$ quantifies the variation of the differential performances results for an evaluated system.

$$\text{Sdiff}_\text{m}(S, lag) = \sigma(\{\text{diff}(S, TC_i, lag) \forall i \in [1, n - lag]\}) \tag{3.9}$$

$\text{Sdiff}_\text{m}(S, lag)$ provides specific information on the differential variation in the performance of a system evaluated in two TCs of *lag* distance. Thanks to this metric, we can compare and understand the possible effect of the distance of the TCs on the performance of the system. A large $\text{Sdiff}_\text{m}(S, lag)$ value represents large changes in the performance of the system evaluated in the TC pairs of *lag* distance.

3.5.1.2 Simulated Evolving Test Collections

We experiment using the simulated ETC defined in section 3.3.4. We use the test collections defined in section 2.5. TREC-Robust (described in section 2.5.2) is composed of 528,155 documents and 250 queries. The documents come from four corpora of news where all the documents are timestamped by the publication day of the news. TREC-COVID (described in section 2.5.1) is composed of 191,160 different documents and 50

queries. The documents are scientific papers related to Covid-19 from CORD-19 dataset [134], where each document is stored with its publication date. As the documents have a timestamp, we will control the evolution of the document set while we consider the full set of queries and the existing relevance assessments $Qrel_i$ for $D_i \times Q_i$.

From a source test collection, we simulate an ETC with a set of parameters according to each configuration (Table 3.4). For the Ov. time-based ETC and Ov. Random ETC, simulations using Robust and TREC-COVID with the defined parameters, we generate 41 epochs. For the Random ETC, we generate the same number of epochs to be able to compare the characteristics of the ETC having the same number of TCs.

		D_i Size	Overlap o	$\mathcal{F}_D$
Evolving	Robust	84504	90%	Date
	TREC-COVID	29576	90%	publish time
Random	Robust	84504	-	-
	TREC-COVID	29576	-	-
Ov. random	Robust	84504	90%	random
	TREC-COVID	29576	90%	random

Table 3.4: Simulation parameters to create DTC based on Robust and TREC-COVID.

For each epoch of the ETC, we evaluate 12 classical information retrieval systems: BM25, PL2, Dirichlet language model and TD_IDF in three versions: with Bo1, KL relevance feedback or without it. All the systems were implemented with default parameters in Terrier v5.2 using the pyterrier library in Python. We evaluate the ETCs using seven performance metrics: precision at 10 first documents (P_10), RPrecision (RPrec), binary preference (bpref), mean average precision (map), normalized discount gain (NDCG) normalized discount gain at 10 first documents (ndcg_cut_10), and reciprocal rank (recip_rank). In a similar way, we use for all metrics the pyterrier implementation.

3.5.1.3 Results

In this section, we present the results of our experiments. We limit our descriptions on the BM25 runs, as we obtain similar behavior for each of the 11 other systems considered. We compare each ETC using the pointwise and pairwise stability. The results describe the differences in terms of stability between our proposed time-based ETC versus Ov. Random and Random ETCs. The latter represents the current state-of-the-art methodology used to analyze the performance of systems in changing environments.

Pointwise stability We evaluate the performance variation of a system evaluated in Evolving, Ov. Random and Random ETCs with pointwise stability metric ($\sigma(x_{S,M})$).

The performance of the system is averaged across all evaluated queries.

Figure 3.6 presents the histograms of the MAP performance distribution for BM25 evaluated in Robust ETC. The performance in the time-based ETC range between 0.2 to 0.28, while the performance in the Ov random and random ETCs ranges between 0.25 and 0.28.

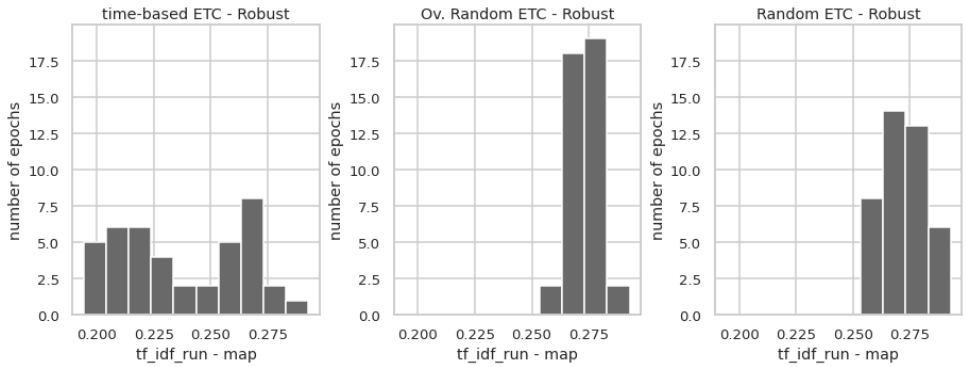


Figure 3.6: BM25 Performance histogram - Robust.

The experiment presents the same tendency for the TREC-COVID BM25 system (Figure 3.7), here the MAP performance of BM25 ranges between 0.1 to 0.30 in the case of an evaluation using the time-based ETC, while the evaluation using Ov. random and random ETCs, the performance is concentrated between 0.15 and 0.25.

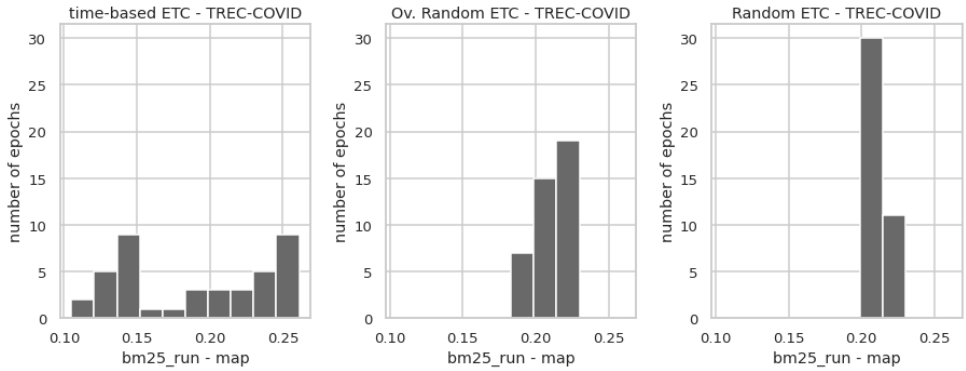


Figure 3.7: BM25 Performance histogram - TREC-COVID.

Table 3.5 extends the results to all the evaluated metrics for Robust ETC. In all the metrics, the std values are larger in a time-based ETC than in a random-based ETC. In terms of performance, all the metrics have the same tendency as MAP metric of figure 3.6. The performance of a system evaluated in Ov. Random and Random ETCs is the same, but if the system is evaluated in an time-based ETC the performance is 10% to

20% smaller. In terms of stability, all metrics used to evaluate the system in the time-based ETC present larger St_m values than Ov. Random and Random ETCs. In the set of metrics, the variation of the performance results in $St_m(BM25)$ is between 2.2 to 4.2 times larger than Ov. Random DTC and Random DTC. In specific, recip_rank presents the most unstable results, with the larger $St_m(BM25)$ for the three ETCs. In this case, the pointwise stability $St_m(BM25)$ is more variable, with time-based ETC 3 times larger than using a Random or Ov. Random ETC.

BM25	Ov. time-based	Ov. random	random
map	0.236 ± 0.027	0.271 ± 0.006	0.271 ± 0.008
bpref	0.214 ± 0.025	0.245 ± 0.006	0.246 ± 0.010
ndcg	0.428 ± 0.040	0.500 ± 0.008	0.500 ± 0.008
ndcg_cut_10	0.316 ± 0.032	0.366 ± 0.008	0.365 ± 0.011
P_10	0.216 ± 0.030	0.244 ± 0.004	0.243 ± 0.007
Rprec	0.228 ± 0.026	0.269 ± 0.006	0.271 ± 0.010
recip_rank	0.460 ± 0.057	0.556 ± 0.018	0.562 ± 0.018

Table 3.5: Pointwise Stability (mean±std) - Robust.

The same tendencies as Robust ETCs are presented in TREC-COVID ETCs (Table 3.6). In TREC-COVID ETCs, the performance mean of BM25 is again between 10% to 20% smaller in a time-based ETC than ov. Random and random ETC. In terms of stability, the St_m values are between 5 to 10 times larger than Ov. Random and Random ETCs. In TREC-COVID, the most unstable metric is P_10, followed by recip_rank.

	Ov. time-based	Ov. random	random
map	0.191 ± 0.052	0.209 ± 0.009	0.211 ± 0.005
bpref	0.314 ± 0.084	0.381 ± 0.010	0.383 ± 0.005
ndcg	0.405 ± 0.076	0.457 ± 0.009	0.458 ± 0.005
ndcg_cut_10	0.378 ± 0.154	0.450 ± 0.020	0.453 ± 0.016
P_10	0.416 ± 0.189	0.505 ± 0.024	0.507 ± 0.016
Rprec	0.226 ± 0.062	0.251 ± 0.009	0.254 ± 0.005
recip_rank	0.591 ± 0.168	0.703 ± 0.019	0.723 ± 0.031

Table 3.6: Pointwise Stability (mean±std) - TREC-COVID.

Pairwise stability The differential performance $\text{diff}_\text{m}(S, TC_i, lag)$, measures the performance change of a system evaluated in TCs pairs of an ETC considering a specific lag distance between the epochs.

Figure 3.8 presents the $\text{diff}_{\text{MAP}}(BM25, TC_i, lag)$ boxplots of BM25 system evaluated in two epochs of the time-based ETC simulated from Robust, with $TC_i \in ETC$ and $lag \in [1, 10]$. Each boxplot presents the distribution of the performance difference of the system evaluated in TC_i and TC_{i+lag}, with lag as the number of TC between TC_i and TC_{i+lag}. As time-based ETC and Ov. Random ETC are built by adding and removing documents with a defined overlap proportion at each epoch, a larger lag value means a smaller overlap ratio between the documents. The lag distance is limited to 10; in $lag = 10$, the TCs do not have any overlapped document. For time-based ETC, the mean diff_{MAP} decreases at each lag value, showing a relation between the lag distance and the performance difference of a system. at the same time, the performance difference is more unstable when the lag distance increases. The results are different for Ov. Random ETC, the mean diff_{MAP} is stable across the lag distances, with more unstable performance difference values when the lag distance increases.

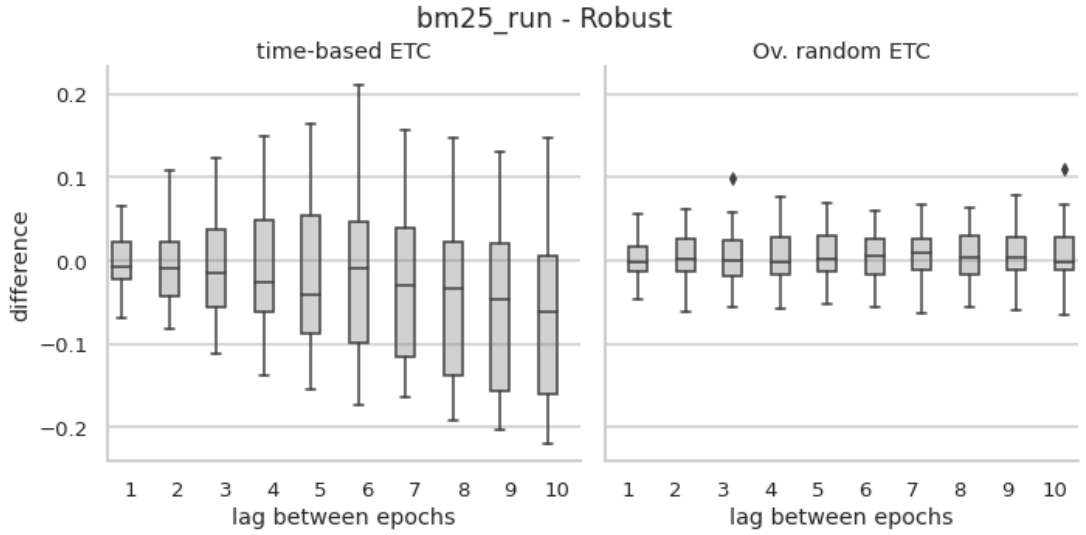


Figure 3.8: $\text{diff}_{\text{MAP}}(BM25, TC_i, lag)$: MAP Performance difference Boxplot of BM25 evaluated on time-based ETC (left) and Ov. Random ETC (right) simulated from Robust.

Figure 3.9 presents $\text{diff}_{\text{MAP}}(BM25, TC_i, lag)$ results for time-based ETC and Ov. random ETC simulated from TREC-COVID. The results follow the same tendencies as Robust ETCs. For time-based ETC, the mean performance difference is related to the lag distance between the epochs (the mean performance difference value increases as the lag is larger), in the line, the performance difference is more unstable when the lag distance is larger. For Ov. Random ETC, the mean diff_{MAP} is stable across the lag distances, with more unstable performance difference values when the lag distance increases.

Table 3.7 extends the results to the seven evaluated metrics for time-based ETC simulated from Robust. In all the metrics, the standard deviation of the differential perfor-

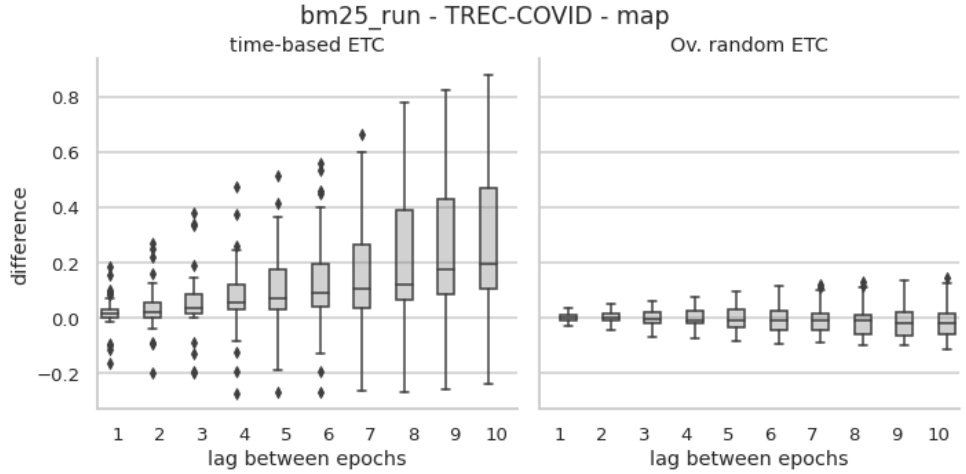


Figure 3.9: $\text{diff}_{\text{MAP}}(BM25, TC_i, lag)$: MAP Performance difference Boxplot of BM25 evaluated on time-based ETC (left) and Ov. Random ETC (right) simulated from TREC-COVID.

mance values increases as the lag distance is larger. The metrics with the most variable differential performance are Rprec, bpref, and MAP, and the metric with the smaller $\text{Sdiff}_{\text{m}}(S, lag)$ in all lag distances is ndcg_10.

lag	map	bpref	ndcg	ndcg_cut_10	P_10	Rprec	recip_rank
1	-0.00±.03	-0.00±.03	-0.00±.02	-0.00±.02	-0.00±.02	-0.00±.03	-0.00±.02
2	-0.00±.04	-0.00±.05	-0.00±.03	-0.00±.03	-0.01±.03	-0.00±.05	-0.00±.04
3	-0.00±.06	-0.00±.07	-0.00±.04	-0.01±.04	-0.02±.04	-0.00±.06	-0.00±.06
4	-0.01±.07	-0.01±.08	-0.01±.05	-0.01±.05	-0.03±.05	-0.01±.07	-0.01±.07
5	-0.01±.08	-0.01±.08	-0.01±.05	-0.02±.05	-0.04±.06	-0.01±.08	-0.01±.08
6	-0.02±.08	-0.02±.09	-0.02±.06	-0.03±.06	-0.05±.07	-0.02±.08	-0.02±.09
7	-0.03±.08	-0.03±.09	-0.03±.06	-0.03±.05	-0.06±.07	-0.03±.08	-0.03±.09
8	-0.04±.09	-0.04±.09	-0.04±.06	-0.04±.06	-0.07±.07	-0.04±.08	-0.04±.10
9	-0.05±.10	-0.05±.10	-0.05±.06	-0.05±.06	-0.09±.07	-0.05±.09	-0.05±.10
10	-0.06±.11	-0.06±.11	-0.06±.06	-0.06±.06	-0.10±.07	-0.06±.10	-0.06±.10

Table 3.7: Pairwise stability (mean diff ± std diff) - Robust.

With TREC-COVID, results presented in Table 3.8, the relation between the lag distance and the differential performance persists for all metrics. With a larger lag distance, the mean and standard deviation of the performance difference increase. In specific for TREC-COVID time-based ETC, the results show larger variability values than for Robust in all metrics. In this experiment, the metrics with the most variable differential

performance are ndcg_10, P_10, and recip_rank, and the most stable metrics are ndcg and map.

	map	bpref	ndcg	ndcg_cut_10	P_10	Rprec	recip_rank
1	0.01±.06	0.01±.05	0.01±.03	0.02±.10	0.03±.12	0.01±.06	0.01±.07
2	0.03±.08	0.03±.08	0.02±.05	0.06±.18	0.07±.23	0.03±.08	0.03±.13
3	0.05±.11	0.05±.13	0.03±.08	0.10±.27	0.12±.34	0.05±.12	0.05±.18
4	0.07±.13	0.07±.17	0.04±.10	0.14±.36	0.18±.48	0.07±.16	0.08±.24
5	0.09±.16	0.09±.20	0.06±.13	0.20±.45	0.25±.60	0.10±.20	0.11±.30
6	0.12±.19	0.12±.24	0.07±.15	0.25±.55	0.32±.72	0.12±.23	0.14±.36
7	0.14±.21	0.15±.27	0.09±.17	0.31±.64	0.41±.84	0.15±.27	0.17±.41
8	0.18±.24	0.18±.30	0.11±.18	0.39±.72	0.50±.93	0.19±.29	0.22±.45
9	0.21±.25	0.22±.32	0.14±.19	0.46±.78	0.59±.01	0.22±.32	0.26±.47
10	0.25±.27	0.25±.33	0.16±.19	0.54±.83	0.69±.08	0.26±.34	0.31±.50

Table 3.8: Pairwise stability (mean diff ± std diff) - TREC-COVID.

3.5.1.4 Conclusion

We have evaluated the pointwise and pairwise stability of a system evaluated in Random ETC, Ov. Random ETC and Ov. time-based ETC. Looking for the configuration that is able to extract more different performances values for a IRS across the epochs of an ETC. The pointwise stability results show that the time-based simulated ETC is able to extract the most different performance values of the IRS in comparison to the Random and Ov. Random ETCs. The pairwise stability, that considers a lag distance between compared epochs, shows that the time-based ETC performance difference is related to the lag difference, being more unstable as the distance between the epochs is larger.

3.5.2 LongEval Evolving Test Collection

As presented in section 3.4, LongEval is composed of three epochs, therefore, three test collections. In this section, we describe the evolution of LongEval, measuring the overlap of elements across the epochs. Then, we compute the pointwise stability and the pairwise stability of a baseline system evaluated in such collections.

3.5.2.1 LongEval Evolution

We present the evolution of LongEval in terms of the number of overlapping documents and queries. Table 3.9 presents the evolution of the number of documents overlapping in epochs of different month distances (lag), June versus July as a lag of one month, July versus September as a lag of two months, and June versus September with a lag of three months. We present the overlap in terms of the number of documents and the ratio between the overlap documents and TC_2 as $\frac{|D_1 \cap D_2|}{D_2}$. As the lag distance between test collections increases, the overlap between the documents decreases at a rate of 2%. Overall the overlap of documents is high (more than 90%) across the epochs.

			overlap	
TC_1	TC_2	months-lag	number	ratio
June	July	1	1.551.617	0.9737
July	September	2	1.033.236	0.9555
June	September	3	1.013.312	0.9370

Table 3.9: Proportion of overlapped documents.

The evolution of the number of queries is presented in Table 3.10. Similar to the overlap of documents, we present the number of common queries in test collections of LongEval and the ratio of overlapped queries in TC_1 and TC_2 as $\frac{|Q_1 \cap Q_2|}{Q_2}$. The ratio of overlapped queries decreases as the lag between the epochs is larger, starting with 27% until 21% of overlapped queries.

			overlap	
TC_1	TC_2	months-lag	number	ratio
June	July	1	240	0.27
July	September	2	214	0.23
June	September	3	190	0.21

Table 3.10: Proportion of overlapped queries.

3.5.2.2 LongEval Evaluation

We evaluate a baseline IRS at each epoch of LongEval and evaluate the pointwise performance stability and the pairwise differential performance in three lag distances. Here BM25 implementation is tested in the June, July, and September English test collections of LongEval[19].

Table 3.11 presents the evaluation of BM25 system using seven performance metrics. First, we report the mean and standard deviation (std) performance of BM25 considering the three epochs. The std describes the pointwise stability of the systems in LongEval. BM25 performance is very stable, with an std ranging between 0.002 to 0.008 for the different metrics with bpref as the most stable, and Rprec with ndcg as the less stable metrics. Afterward, we present a pairwise differential performance of the system. In this case, we consider the three lag distances again: June versus July (lag=1), July versus September (lag=2), and June versus September (lag=3). As we have one measurement for each lag we can not compute the pairwise stability, instead we report the pairwise differential performance, that measures the relative difference between the performance of the IRS evaluated in TC_1 with respect to TC_2. The performance difference increases with more lag distance in the case of Rprec, ndcg_cut_10, P_10. The performance difference is larger considering June and September (lag=3) than June and July (lag = 1), for all the metrics excepting bpref.

	Stability	Differential Performance		
		June - July	July - Sept	June - Sept
Metric	mean±std	lag=1	lag=2	lag=3
map	0.16±0.005	0.043	0.031	0.076
Rprec	0.12±0.008	0.009	0.108	0.118
bpref	0.32±0.002	-0.013	0.009	-0.004
recip_rank	0.26±0.005	-0.014	0.042	0.027
P_10	0.10±0.006	0.012	0.104	0.117
ndcg	0.29±0.008	-0.002	0.050	0.048
ndcg_cut_10	0.18±0.005	0.019	0.033	0.053

Table 3.11: Systems Performance stability (mean±std) and Performance Difference BM25 system.

[19]French and English baseline performance values are reported in LongEval resource paper [50], proving that English results are correlated to the original datasets in French

3.5.2.3 Summary

In this section, we presented first experiments on LongEval test collection. We evaluate a baseline system in the different test collections that composes LongEval ETC. First, we observed the evolution of the ETC in terms of documents and queries, with high overlaps in the case of documents and smaller overlaps in the case of queries. Second, we computed the pointwise stability and the differential performance of the baseline in LongEval. The performance is stable across the test collections with less than 0.01 of standard deviation. In the same line, the performance difference of IRS considering two epochs, is close or less than 10% in all metrics.

3.6 Conclusion

In summary, in this chapter we present and formalize Evolving test collections. We propose a methodology to simulate a ETC from a Static Test Collection and the construction of a real Evolving test collection by data acquisition using a commercial search engine (Qwant). Using the simulation methodology we propose three different ETC based on random and temporal features, that are supposed to mimic real cases of evolution of the different components of a test collection as the queries and documents.

In the experiments section, we proved that an time-based ETC is more useful for evaluating the system to extract different performance values. With respect to the performance change of an IRS across the epochs of an ETC, we did obtain different behaviour for time-based versus Ov. Random and Random simulated ETCs in the stability dimensions. The time-based ETC presented more variable results using any metrics, such as variability increase if we compare the performance of a system in more *lag*-distances epochs of the ETC.

We proposed a framework to continuously acquire data to create an evolving test collection. We applied the framework to create LongEval, an evolving test collection with documents, queries and relevance judgments that change across epochs. Results show that LongEval is evolving in terms of components and performance evaluation.

Using time-based ETC and LongEval ETC we are able to explore the evolution of an IRS system performance across changing test collections. In this chapter, we evaluate the performance stability of a system across the epochs of the ETCs. Nevertheless, there is no framework to compare the performance of the systems across the epochs.

Chapter 4

Evaluating Systems on Evolving Test Collections

In this chapter, we aim to compare the performance of one or several systems evaluated on evolving test collections. To achieve this goal, we propose a continuous evaluation framework that quantifies differences in system performances across the ETC. We introduce the continuous evaluation and our continuous result delta evaluation framework in section 4.1. In section 4.2, we formalize the performance difference quantification in the form of result deltas. In section 4.3 we detail the continuous evaluation framework for systems evaluated in evolving test collections. Section 4.4 presents the experiments that validate our proposal, and section 4.5 discusses the results of the experiments. Finally, in section 4.6 we present the conclusions of our proposal.

4.1 Introduction

The goal of a continuous evaluation of IR systems is to provide information about the quality of systems in a repeated evaluation using evolving test collections. This information shows the variability of the performance of one or several systems and how their quality change in different contexts. We propose here a continuous evaluation framework to evaluate accurately one or several systems by taking into account such evolutions. Our goal is to support a longitudinal evaluation of systems across the data changes.

Our continuous evaluation framework is built upon the specific features of an evolving test collection (ETC). As defined in Chapter 3, an ETC is composed of a sequence of test collection at different *epochs*, which components (documents, queries, and relevant assessments) gradually changes over time. As the changes are supposed to be gradual, we do not expect radical changes of the collection components along the evolution. We take benefit from this hypothesis when defining our evaluation framework on such ETCs.

The performance of a system changes when it is evaluated in different collections [45].

In a continuous evaluation, *one system* may be evaluated in an ETC. To measure the effect of evaluating such a system in different test collections of an ETC, a meta-analysis is performed (see section 2.2.3). It provides an overall evaluation of one system versus a baseline across several evaluations.

The main challenge of a continuous evaluation arises when the task is related to comparing the performance of *different systems evaluated in different collections* of an ETC. Standardization (described in section 2.3) presents a method to transform the performance of a system considering the topic difficulty. We make use of such standardization to measure the performance difference across test collections. We consider not only the difficulty of the topics but the difficulty of the test collection when computing the performance difference of systems evaluated in an ETC.

Overall, we propose in this chapter a continuous evaluation framework that relies on evolving test collections to measure the performance differences among evaluated systems. Our framework introduces Result Deltas ($\mathcal{R}\Delta_M$) as a formal definition of the performance difference measured with a metric M.

Our evaluation framework consists of three key steps, each contributing to a comprehensive assessment of system performance:

1. Comparability Validation: We establish criteria to determine when it is meaningful to compare systems across two epochs. This allows for valid comparisons between systems.

2. Performance Comparison Strategy: In the second step, we define a strategy for comparing the performance of the evaluated systems. This strategy relies on a set of baseline systems. In this step, we establish a robust framework for systematically assessing and comparing the performance of different systems.

3. Longitudinal Evaluation: In the final step, we conduct the evaluation of the test systems. This involves analyzing their performance on the compatible test collections, taking into account the transformed values from the previous step to compare the systems by $\mathcal{R}\Delta_M$ values.

By following this three-step continuous evaluation framework, we ensure the comparability of test collections epochs, establish a meaningful performance transformation for comparison, and conduct a comprehensive evaluation of system performance as a longitudinal analysis.

4.2 Definitions

The evaluation of IR systems uses a test collection, as presented in chapter 2. In a Web Search Engine, modifications of the IRS are constantly incorporated into the Web search

engine, leading to the evaluation of different systems on different test collections (i.e., each new IRS version is evaluated in the last version of the ETC). We define continuous evaluation as:

Definition 9 *A Continuous Evaluation is a task of comparing the performance of one or several systems in an evolving test collection.*

In a Continuous evaluation framework, it is essential to address the changes occurring across the epochs of the ETC and the modifications made to the IR system itself, as these factors can significantly impact the evaluation of a system measured by a performance metric M. The question of being able to measure accurately the impact of the modifications of the test collection, as well as the systems, on the evaluation results must be tackled.

To compare the performance of systems evaluated continuously, using a performance metric M, we propose **Result Deltas** ($\mathcal{R}\Delta_M$), defined as:

Definition 10 *A Result Delta, $\mathcal{R}\Delta_M$, is the measurable difference performance, given a performance metric M, between two systems evaluated on two successive epochs of an ETC, where the systems and the test collection components could evolve.*

Considering that an ETC consists of several epochs ($TC_i \in ETC$), we propose a pairwise comparison of systems' performance. Figure 4.1 illustrates this continuous evaluation, showing three different systems ($S = \{S1, S2, S3\}$) evaluated on an ETC composed of three epochs ($ETC = [TC_1, TC_2,, TC_3]$). Here $M(TC_i, s_j)$, with $TC_i \in ETC$ and $s_j \in S$, represents the global performance (described in section 2.2.2) of a system s_j evaluated in TC_i using the metric M.

As the systems are evaluated within different epochs, the comparison takes into account the performance of systems assessed on different test collections, and relying solely on absolute metric values, as described in section 2.2.2.1, is not feasible. Therefore, we propose to build an evaluation framework to compute evaluation differences as $\mathcal{R}\Delta_M$, comparing $S1$ evaluated in TC_1.

We formalize $\mathcal{R}\Delta_M$ (Equation 4.2): considering two systems (S_1 and S_2), two ETC epochs (TC_1, TC_2) and one metric M that is used to measure the performance of S_1 evaluated using TC_1 and S_2 evaluated using TC_2, $\mathcal{R}\Delta_M$ returns a real value estimating a performance difference:

$$\mathcal{R}\Delta_M : ((TC_1, S_1), (TC_2, S_2)) \rightarrow \mathbb{R} \tag{4.1}$$

As defined, in a continuous evaluation, the systems and TCs could evolve along the epochs. We define three kinds of $\mathcal{R}\Delta_M$ that can be measured according to the element that may change (the system or the TC):

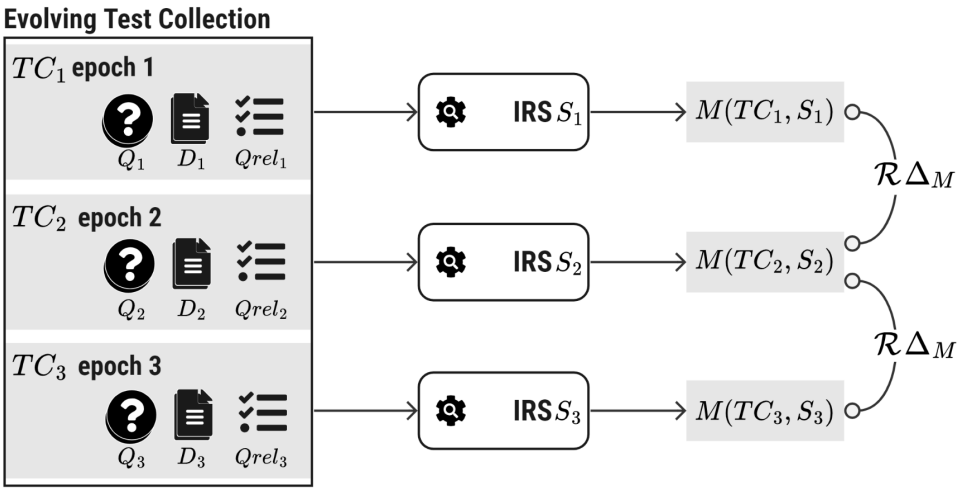


Figure 4.1: Continuous Evaluation based on Result Deltas, with M as the performance measure of an IRS evaluated in a TC epoch of the ETC.

- $\mathcal{R}_s\Delta_M$: When we have two different IR systems evaluated in the same epoch, as a classical IR evaluation.
- $\mathcal{R}_e\Delta_M$: When the same IR system is evaluated in two epochs.
- $\mathcal{R}_{se}\Delta_M$: When both epochs and systems are different.

In figure 4.2, we present the different kinds of $\mathcal{R}\Delta_M$. To illustrate $\mathcal{R}_e\Delta_M$, $\mathcal{R}_s\Delta_M$ and $\mathcal{R}_{se}\Delta_M$, we address the case where system S_1 (black dots) is evaluated in TC_1 and TC_2 and S_2 (gray dots) is evaluated in TC_2 and TC_3. The simplest case of $\mathcal{R}_s\Delta_M$ applies when the systems are different and they are evaluated in the same epoch ($TC_1 = TC_2$). In such case, $\mathcal{R}_s\Delta_M$ measures the performance change of one system against another one in the same conditions. This case is similar to a classical Evaluation of IR systems, i.e., comparing a system to a baseline. The second case of $\mathcal{R}_e\Delta_M$ considers the same IR system ($S_1 = S_2$) evaluated in two epochs. For example, $\mathcal{R}_e\Delta_M$ can be implemented as a reproducibility metric presented in section 2.4.3, as their goal is to measure the performance difference of the same IRS in different test collections. The last type of $\mathcal{R}_{se}\Delta_M$ is measured between different systems in different epochs. In such a case, $\mathcal{R}_{se}\Delta_M$ depends on the evolution of the systems and also on the evolution of the test collection.

Our goal is to compute $\mathcal{R}_{se}\Delta_M$, but it can hardly be measured, as the two systems are not directly comparable: both the ETC epochs and the systems are different. To get an estimation of this measure, we propose:

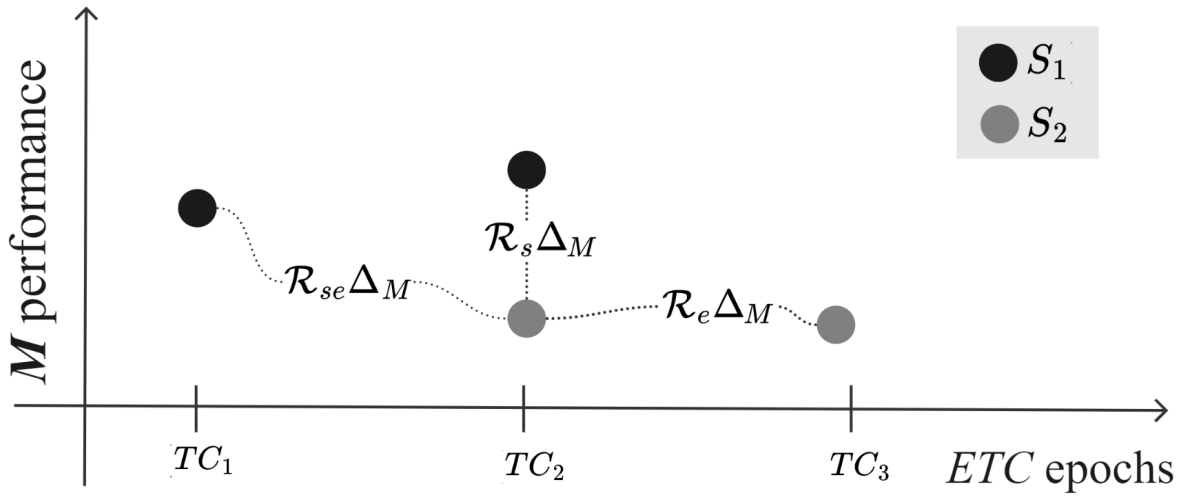


Figure 4.2: $\mathcal{R}\Delta_M$ of systems S_1 and S_2 evaluated in TC_1, TC_2 and TC_3, three ETC epochs.

Definition 11 *A continuous result delta evaluation framework that defines how to compute $\mathcal{R}_{se}\Delta_M$ from a longitudinal analysis of transformed performance values of systems evaluated in different epochs of an ETC.*

The Continuous evaluation framework should define when comparing systems evaluated in different epochs makes sense. Then how to transform the performance values of all the systems evaluated in different epochs to be able to compare them into a longitudinal analysis. Using our illustrated example in Figure 4.1, we need to compare the performance of S_1 and S_2 considering TC_2 and TC_3. As S_1 is not evaluated in TC_3, the task of the framework is to make both performance values comparable. Using this comparable performance value, it is possible to compute $\mathcal{R}_{se}\Delta_M$ and conclude which system outperforms the other.

4.3 Continuous Result Delta Evaluation Framework

We present now our framework: *Continuous Result Delta Evaluation*. First, we present the research questions and hypothesis that define the characteristics of the framework in section 4.3.1. In section 4.3.2, we describe an overview of the three steps that compose the framework: (i) the comparability validation in section 4.3.3, (ii) the performance comparison strategy in section 4.3.4, and (iii) the longitudinal evaluation in section 4.3.5.

4.3.1 Research Questions and Hypotheses

In a Continuous Evaluation we are interested in comparing systems evaluated in different test collections, then we focus on two main research questions:

RQ1 How to correctly conclude which system is better considering all the evaluation epochs?

RQ2 How to compare the performance of systems with respect to a specific evaluation epoch?

To answer these questions, our Continuous Result Delta Evaluation Framework is based on the characteristics of an Evolving Test Collection. Here we assume that the gradual changes in the evolving test collection can be measured using simple features (overlapping documents, overlapping topics, etc.). In Chapter 3, we presented a simulation method to build evolving test collections using such simple features. Then, we define what are the expected impacts of a gradual evolution of a collection on the behavior of systems so that we may bind our study to meaningful cases. As the evolving test collection considered is gradually changing, we suppose that:

H1 If we run the same set of information retrieval systems at successive epochs, a meaningful evaluation of systems supposes that the Ranking of these Systems (RoS), according to the same evaluation metric, stays the same. Such an assumption is similar to the work of Soboroff [115] when no change occurs between successive epochs.

H2 As the epochs change gradually, we assume that there exist some features of the epochs which may be used as grains, using which we can study the (un-)expected changes in the evaluation of the systems. In this framework, we will study grains defined as clusters of topics in each successive epoch considered.

4.3.2 Overview

With the hypotheses listed above and the definition of the continuous result delta evaluation, we are able to propose an evaluation framework using evolving test collections. Our proposed framework relies on three steps processed in sequence (in Figure 4.3):

Step 1. The Comparability Validation (CV) is a decision step that detects if successive epochs are *comparable*, i.e., if the differences between them are not too large according to our continuous evaluation hypothesis $H1$ (defined in section 4.3.1).

Step 2. A Comparison Strategy (CS) step, in which we define the strategy to transform the system's performance into a *common scale*. It can be implemented by (2.a) a *Pivot* Comparison, by (2.b) a *Projection* comparison; and by (2.c) *Grains* comparisons:

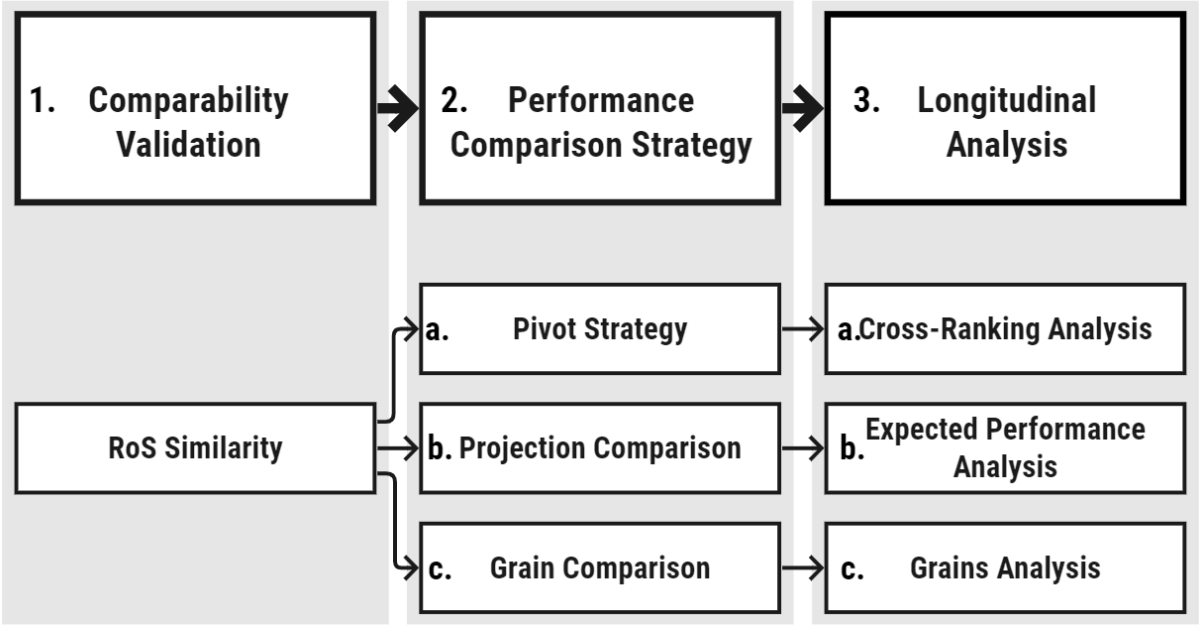


Figure 4.3: Steps of the Continuous Result Delta Evaluation Framework.

2.a a **Pivot Strategy** (PIV) defines a system as a pivot to create a ranking of systems by incorporating the evaluation of the pivot across the compared epochs, supported on hypothesis $H1$ (defined in section 4.3.1);

2.b a **Projection** (PRO) step defines projection functions to transform the performance of a system from one epoch to another one. The functions are computed using a set of baselines and a set of queries that are common across the epochs, using hypothesis $H2$ (defined in section 4.3.1); or,

2.c a **Grain Definition** (GRA) step defines grains of queries across the epochs based on a grouping feature, to compute a standardized performance in such grains, using hypothesis $H2$ (defined in section 4.3.1).

Step 3. A Longitudinal Analysis (LA) step that allows the continuous evaluation of systems along several successive epochs. Depending on the method used, a specific analysis is performed, from which $\mathcal{R}_{se}\Delta_M$ can be computed:

a. **Continuous-Ranking Analysis** compares the pivot versus the tested systems to create a ranking of systems at each pair of epochs.

b. **Expected Performance Analysis** step calculates an expected performance using the projection of a system in a previous epoch, and it is contrasted to the last evaluated epoch.

c. a **Grains Analysis** compares the performance of systems by *meaningful grains* of different queries of two epochs.

Therefore, the Continuous Result Delta Evaluation is formed of three steps, resulting in a list of performance values for each evaluated system across ETC. Therefore, $n-1$ $\mathcal{R}_{se}\Delta_M$ values can be computed, one list for each pair of tested systems.

$$\text{Cont}\mathcal{R}\Delta_M\text{Eval} : (CV, CS, LA, ETC) \rightarrow \mathbb{R}^n \tag{4.2}$$

where each step CV, CS and LA has a set of respective parameters according to the specific method described, which are detailed in the following sections.

Our framework is expected to exhibit precise indications of the behavior of one or several systems over an evolving test collection under strict conditions that lead to meaningful results: we assess that epochs can be compared, we evaluate the corrected of the pivot strategy, and we assess that the granularity used on which we evaluate the systems is also valid.

4.3.3 Comparability Validation

The **Comparability Validation (CV)** seeks to assess if it is meaningful to compare systems across two epochs: if the epochs are too different, we cannot draw any useful conclusions about the evolution of the evaluation metrics.

We follow the same principle as the validation of test collection *reusability*, that defines a test collection to be reusable if and only if it can be used for precise measurements of the performance of systems that did not contribute to its judgments [26]. The validation procedure tests if new runs are ranked in the same order as a collection that considers its contributions to the set of judgments [39]. This supports the claim that the collection is reusable.

The CV step assumes that there exists a set $S = \{s_i\}$ of IR Systems s_i that are tested over each successive epoch considered. It is then possible to rank these systems over each epoch.

Comparability is defined as [122]:

> **Definition 12** *Two epochs TC_1 and TC_2 of an ETC are comparable according to an evaluation measure M, if for a given set S of IR systems, the ranking of the systems in S according to M is the same in TC_1 and TC_2.*

This step uses, as in [26, 39], Kendall τ coefficients over the ranking of systems at successive epochs. The parameters of the CV step are: the evaluation metric considered (e.g., MAP), a threshold τ_{CV} that supports the decision of two rankings of systems are similar enough, and the set of systems S: in such case, we decide that the epochs are *comparable*:

$$CV = (S, M, \tau_{CV})$$

As a reference, a threshold τ_{CV} greater than 0.9 implies two rankings of systems that can be considered equivalent, and a threshold τ_{CV} less than 0.8 denotes rankings with noticeable differences [20, 128].

4.3.4 Comparison Strategy

We present now three strategies to compare systems evaluated in an evolving test collection. First, we are interested in answering *RQ1* by computing a continuous ranking of systems, that considers the evaluation of different systems in several epochs. To achieve this goal, we propose a Pivot strategy (in section 4.3.4.1) that uses a reference system to rank systems evaluated using different test collections. Second, we propose a Projection comparison (in section 4.3.4.2) focused on answering *RQ2*, with the goal of comparing the performance of systems in a specific epoch of interest. The Projection comparison uses a set of reference systems as baselines to compute standardization functions and then project the system's performances that are evaluated using the same set of queries. Third, we propose a Grain Comparison (in section 4.3.4.3) with the goal of comparing the standardized performance of systems at a specific grain, which is defined by different sets of queries.

4.3.4.1 Pivot Strategy

A Pivot (PIV) Strategy [57] aims to compare systems measuring their improvement versus a baseline system, called Pivot. It is used to create a *continuous* ranking of systems evaluated on different test collections. We take inspiration from reproducibility metrics and the meta-analysis technique (see section 2.4.3) to extend them to be applied to compare several systems in an evolving test collection.

The key point of this strategy lies in the choice of the pivot. First, we define how to apply the Pivot strategy with a known Pivot, and then we define how to select a pivot from a set of reference systems.

Ranking of systems using a Pivot: To rank the systems, we compute a $\mathcal{R}_{se}\Delta_M$ value for two evaluated systems. We propose to estimate $\mathcal{R}_{se}\Delta_M$ using a reference system called **Pivot system** (S_{pivot}), which would be evaluated within the two epochs considered. First, $\mathcal{R}_s\Delta_M$ is computed between each system and the pivot within each TC considered. Then, both $\mathcal{R}_s\Delta_M$ are used to compute $\mathcal{R}_{se}\Delta_M$ and compare the two systems over the two epochs of the ETC. Figure 4.4 illustrates the Pivot Strategy considering two epochs of an ETC, TC_1 and TC_2. Here the pivot system S_{pivot} is evaluated in both epochs.

The epochs TC_1 and TC_2 are comparable, as defined in the previous section 4.3.3. This means that the performance of the systems could change, but their order (sorted by decreasing performance) persists. We take advantage of this situation to create an RoS from $\mathcal{R}_{se}\Delta_M$. In this case, the pivot system will help us to relate the systems across

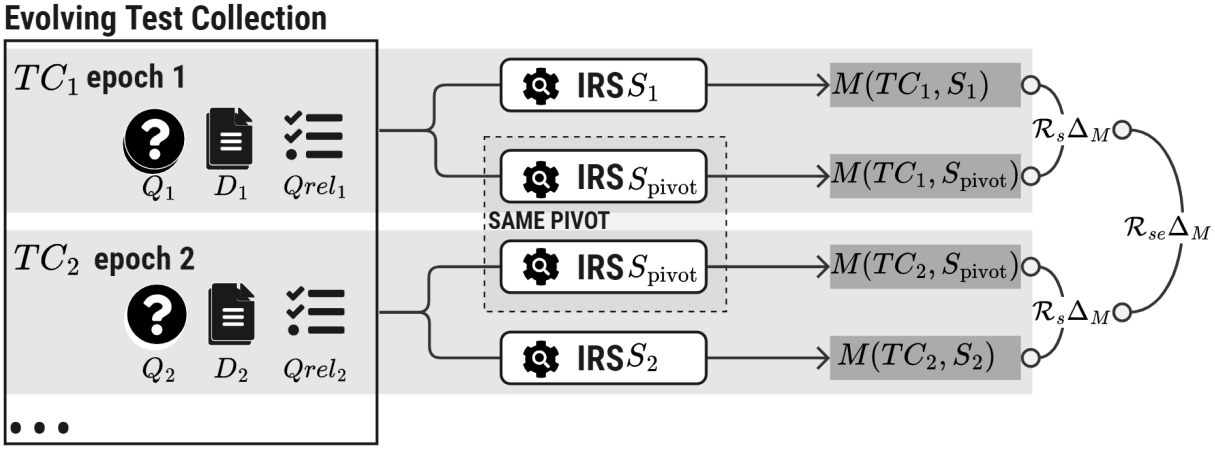


Figure 4.4: Pivot Strategy to compute $\mathcal{R}_{se}\Delta_M$.

the epochs comparing by comparing the performance of S_1 ($M(TC_1, S_1)$) with the performance of the S_{pivot} ($M(TC_1, S_{\text{pivot}})$) as $\mathcal{R}_s\Delta_M((TC_1, S_{\text{pivot}}), (TC_1, S_1))$ and $M(TC_2, S_2)$ with $M(TC_2, S_{\text{pivot}})$ as $\mathcal{R}_s\Delta_M((TC_2, S_{\text{pivot}}), (TC_2, S_2))$.

First, $\mathcal{R}_s\Delta_M$ is measured using the relative distance between the pivot system and the evaluated system S_1:

$$\mathcal{R}_s\Delta_M((TC_1, S_1), (TC_1, S_{\text{pivot}})) = \frac{M(TC_1, S_1) - M(TC_1, S_{\text{pivot}})}{M(TC_1, S_{\text{pivot}})}$$

in the same way, the relative distance between the pivot and the S2 is computed, both systems evaluated in TC_2, using $\mathcal{R}_s\Delta_M((TC_2, S_2), (TC_2, S_{\text{pivot}}))$.

Based on this, $\mathcal{R}_{se}\Delta_M$ relates two systems evaluated in two different epochs as:

$$\begin{aligned}\mathcal{R}_{se}\Delta_M((TC_1, S_1), (TC_2, S_2)) =& \mathcal{R}_s\Delta_M((TC_2, S_2), (TC_2, S_{\text{pivot}})) \\ &- \mathcal{R}_s\Delta_M((TC_1, S_1), (TC_1, S_{\text{pivot}}))\end{aligned}$$

With $\mathcal{R}_{se}\Delta_M$ computed from $\mathcal{R}_s\Delta_M$ we can rank S_1 and S_2. As an illustration, if $\mathcal{R}_s\Delta_M((TC_1, S_1), (TC_1, S_{\text{pivot}}))$ is greater than $\mathcal{R}_s\Delta_M((TC_2, S_2), (TC_2, S_{\text{pivot}}))$, i.e., $\mathcal{R}_{se}\Delta_M((TC_1, S_1), (TC_2, S_2)) < 0$, then S_1 is greater than S_2 in both epochs:

$$\begin{aligned}&\mathcal{R}_s\Delta_M((TC_1, S_1), (TC_1, S_{\text{pivot}})) > \mathcal{R}_s\Delta_M((TC_2, S_2), (TC_2, S_{\text{pivot}})) \\ &\implies \mathcal{R}_{se}\Delta_M((TC_1, S_1), (TC_2, S_2)) < 0 \\ &\implies M(TC_1, S_1) > M(TC_1, S_2) \wedge M(TC_2, S_1) > M(TC_2, S_2)\end{aligned} \tag{4.3}$$

Pivot Selection strategy: To select a pivot from a list of possible candidates, we assess the quality of each of them. We select the S_{pivot} that computes the most *correct* Ranking of Systems (RoS), with respect to a reference RoS. To evaluate the quality of a pivot we compare the following (In figure 4.5):

- RoS_{ref} a reference RoS according to a specific epoch of reference (TC_1) as the ground truth. All the systems in the set of reference systems are evaluated in this epoch.
- RoS_{pivot} is built using the S_{pivot} from a split of the reference epoch (TC_1 is split in $TC_{1,1}$ and $TC_{1,2}$). RoS_{pivot} uses the result deltas of the pivot under consideration. In each split half of the systems are evaluated.

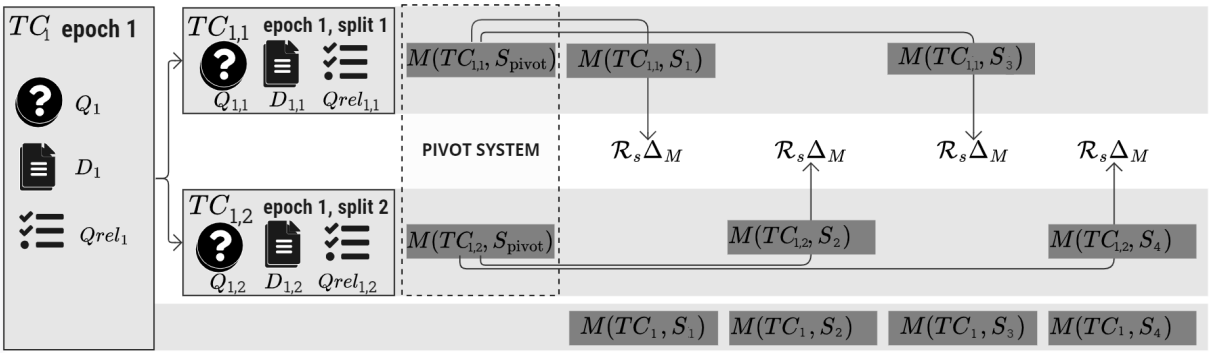


Figure 4.5: Pivot selection with TC_1, $TC_{1,1}$ and $TC_{1,2}$.

If the two rankings are the same, this means that the pivot is able to support the indirect comparison of systems correctly. To evaluate the correctness of a pivot, we measure the similarity between RoS_{pivot} and RoS_{ref} (e.g. using Kendall's Tau similarity metric). The correctness of a pivot must be compared to a baseline. To do that, we define a $RoS_{baseline}$ that is constructed under the same TC epochs created for the RoS_{pivot}. The $RoS_{baseline}$ orders the absolute performance values of the two system sets evaluated on each TC epoch. Then, we measure the similarity between the $RoS_{baseline}$ and RoS_{ref}. We expect higher similarity values using the pivot strategy than with the absolute performance values.

This step needs a set of systems from which to select a pivot system as $S = s_i$ taking one system as the candidate Pivot and the others as the ranked systems, according to an evaluation metric M. Therefore the parameters of the PIV step are:

$$PIV = (S, M)$$

As a summary, the Pivot Strategy is useful for comparing systems evaluated in two different epochs. As a result, the pivot strategy is able to indicate which system is assumed

to outperform the other across the epochs of the ETC. The pivot strategy relies on one reference system, which may be a limitation in the evaluation of different systems across time.

4.3.4.2 Projection Comparison

In this strategy, we consider several reference systems to compare the performance of systems evaluated in different epochs of the ETC.

In a projected (PRO) comparison strategy, we calculate $\mathcal{R}_{se}\Delta_M$ by projecting values to compare systems evaluated at different epochs using the same queries. This process involves comparing the performance of the current system with a projected performance of a system evaluated in a previous epoch, considering the most recent epoch as the reference point. The projection results in the expected performance of a system, which is determined using a set of reference systems.

By applying such projection, we can estimate the $\mathcal{R}_{se}\Delta_M$ measure of two systems (S_1 evaluated in TC_1 and S_2 evaluated in TC_2) with respect to a specific epoch of interest (TC_2). This estimation is computed by comparing the performance of S_2 ($M(TC_2, S_2)$) with the projected performance of system S_1 as $M_{proj}(TC_2, TC_1, S_1)$). Here, M_{proj} is the projection of the performance of S_1 evaluated in TC_1 into the current one (TC_2). Using these elements, $\mathcal{R}_{se}\Delta_M$ is computed as:

$$\mathcal{R}_{se}\Delta_M(TC_1, S_1, TC_2, S_2) = M(TC_2, S_2) - M_{proj}(TC_2, TC_1, S_1)$$

The projection comparison consists of two main steps. First, we compute a standardization function for each query, which transforms the performance values for each query according to the difficulty of the query for the set of reference systems. This step ensures that all systems' performance values are represented on a *standardized* scale. Secondly, we perform a projection step, where the performance of the system is transformed from the *standardized* scale to the *current epoch* scale. This step allows for a meaningful comparison of the systems evaluated in different epochs using performance values in terms of an epoch of interest.

The **standardization step** transforms the systems performance values to a scale of 0 to 1 . As we know from step CV that the epochs are comparable, it makes sense at this step to standardize the evaluation metrics (e.g., we note std(MAP) as the standardized value of the MAP evaluation metric). As seen in section 2.4, standardization could be parametric when a distribution of the data is assumed [99, 136], or non-parametric as in the case of the empirical standardization proposed by Urbano [123]. One-step standardization uses a cdf to transform the raw performance score x into a standardized score $y = F(x) = P(X \leq x)$. The function $F(x)$ is estimated using the performance scores of a set of sample systems based on a normal, linear, or empirical distribution.

We compute standardization functions $\hat{F}(x)$ for each query q at each TC epoch, using

the performance metric M with respect to a set of reference systems S, as $std_{q,TC,M,S}()$. This function normalizes the performance of a system with respect to the set of reference systems standardized by query difficulty.

In the **projection step**, we use the standardization functions defined in the previous step ($std_{q,TC,M,S}$) to use the inverse of the standardization function ($std^{-1}_{q,TC,M,S}$) which projects the performance value from the standard scale (0,1) to the scale of TC epoch.

More precisely, for a system evaluated on epoch TC_1 that has a M performance x in q_1, we use functions $std_{q_1,TC_1,M,S}$ and $std_{q_1,TC_2,M,S}$, to define $proj_{q1,TC_1,TC_2,M,S}$ as:

$$proj_{q1,TC_1,TC_2,M,S}(x) = std_{q_1,TC_2,M,S}(std_{q_1,TC_1,M,S}(x))^{-1}$$

Notice that the projection function receives performance values in the scale of the performance metric M and returns as an image in the same scale of metric M[1]:

$$proj_{q,TC_1,TC_2,M,S} : M \rightarrow M$$

Figure 4.6 exemplifies the projection of the performance. Suppose that

- the evaluation measure M for the query q_1 is equal to 0.7 in epoch TC_1 ;
- the standardization function for q_1 in TC_1 gives $std_{q_1,TC_1,M,S}(0.7) = 0.9$;
- the standardization function for q_1 in TC_2 gives $std_{q_1,TC_2,M,S}(0.6) = 0.9$.

The projection of the evaluation measure value 0.7 of q_1 in TC_1 to TC_2 is equal to $proj_{q,e1,e2,M,S}(0.7) = 0.6$. We can then compare the actual value of the system for the query q_1 in TC_2 with its projection from TC_1.

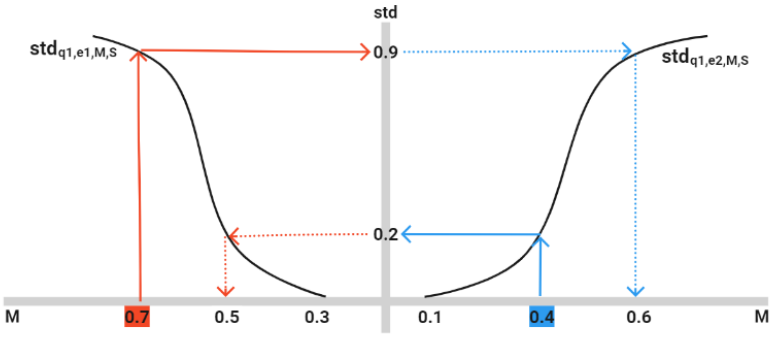


Figure 4.6: Diagram of projection of values (in blue (resp. in red) the standardization function in TC_1 (resp. TC_2).

[1]We consider global performance values and metrics in a cardinal scale of real values.

To compute $M_{proj}(TC_2, TC_1, S_1)$ we consider the projection all the common queries in TC_1 and TC_2:

$$M_{proj}(TC_2, TC_1, S_1) = \frac{1}{|Q_1 \cap Q_2|} \sum_{q_i \in Q_1 \cap Q_2} proj_{q_i, TC_1, TC_2, M, S}(x_i)$$

$$\text{with } x_i = M(q_i, S_1)$$

As noticed, in an ideal case, these functions are both bijections; otherwise, the projection can not be computed. To deal with this issue, we propose an empirical inverse function:
(1) if $std_{q,TC_1,M,S}(x) \geq std_{q,TC_2,M,S}(1)$ then:

$$std_{q,TC_2,M,S}(x)^{-1} = 1$$

(2) If $std_{q,TC_1,M,S}$ is not injective, we compute two inverse functions:

$$(2.1)\ std_{q,TC_2,M,S}(y)^{-1} = min(\{x | std_{q,TC_1,M,S}(x) = y\})$$

$$(2.2)\ std_{q,TC_2,M,S}(y)^{-1} = max(\{x | std_{q,TC_1,M,S}(x) = y\})$$

Finally, we define a range of minimum and maximum projected scores using two projections. In this case, two M_{proj} values are computed, the first as the minimum projection mean and the second as the maximum projection mean.

Overall, the PRO step has two parameters: the set of reference systems S, the evaluation metric considered M, and the standardization method to be considered.

$$PRO = (S, M, method_{std})$$

As the corpus relies on the non-common query sets, we propose a method that supports the comparison using different query sets and different reference systems.

4.3.4.3 The Grain Comparison

The grain (GRA) comparison step is used to compare the performance of systems evaluated on possibly different query sets. A granularity defines excerpts from the full data of one epoch. The goal of a grain definition is to compare the performance of systems evaluated in different epochs by clustering the behavior on similar queries, even if the queries are not the same. We analyze the (un-)expected changes in the evaluation of the systems in these clusters.

A grain comparison consists of two steps. Firstly, a standardization method is applied to ensure that the performance of systems evaluated in different epochs is represented on the same scale. This step follows the same definition described in the PRO step. In this

case, standardization helps in aligning the performance values for meaningful comparison. Secondly, a grain definition where a set of grains $G = \{g_i\}$ is defined to evaluate the systems across the epochs. These grains capture specific subsets of queries that exhibit similar characteristics or behavior. By comparing the standardized performance of systems within these grains and a metric M, we can compare their performance and observe any patterns or trends across the different epochs.

A grain comparison proposes to compute the $\mathcal{R}_{se}\Delta_M$ relying on each standardized grain performance measurement (M_{g_i}), as follows:

$$\mathcal{R}_{se}\Delta_M((TC_1, S_1), (TC_2, S_2)) = M_{g_i}(TC_2, S_2) - M_{g_i}(TC_1, S_1)$$

The definition of a set of grains relies on a subset of queries that can be used to exhibit the performance of a system [27, 60]. To create a set of grains we may group the queries by similar performance according to a set of reference systems. With these similar-performing grains, we can analyze specific behaviors on similar queries on evolving collections.

We propose to validate the defined granularity so that it provides added value compared to the full epochs comparisons. The validation of granularity is similar to what is done in the comparability validation, as it relies on the systems in S, but is computed independently on the respective grains over successive epochs. Such validation also assumes a threshold τ_{GRA}, which checks if the granularity is valid. Then, explanations of the changes in evaluation metrics of the IR system considered can be described according to the granularity.

This GRA step has five parameters: the set of reference systems S, the evaluation metric considered (M), the definition of the grains ($G = g_i$), the standardization procedure ($method_{std}$), the decision threshold τ_{GRA} above which the granularity is comparable across the successive epochs. Therefore the GRA step is defined as:

$$GRA = (S, M, method_{std}, \tau_{GRA})$$

We propose three comparison strategies, each of them with a specific number of constraints (summarized in Table 4.1). The PIV comparison has the condition of constantly using the same reference system as Pivot. If we apply another reference system, we cannot create an RoS using previous epochs. The PRO strategy does not have this constraint because the standardization functions can be computed using different sets of systems. However, the PRO strategy needs a common set of queries to compute the projection functions. Therefore, if a query is removed from the test collection, then it is excluded from the projection function computation. Finally, we propose the GRA strategy that does not have any of these constraints, as it relies on standardized values, it can use different sets

of reference systems across epochs, and as the grains are defined in terms of a grouping feature, the queries that compose each grain may change across epochs.

Strategy	Systems	# Systems	Topics
PIV	constant	1	different
PRO	different	several	same
GRA	different	several	different

Table 4.1: Comparison Strategy constraints.

4.3.5 Longitudinal Analysis

The Longitudinal analysis step (LA) describes the behavior of tested system S_t, and compares the performance of two systems $s_{t1}, s_{t2} \in S_t$ across all the epochs considered in ETC to perform a Continuous Evaluation. We remind you that our framework is dedicated to assessing the behavior of systems S_t on the considered evolving test collection, which is composed of several epochs. LA, in general terms, does the following:

1. runs s_t on the epochs that are comparable according to CV,
2. presents the *longitudinal* behavior of s_t that depicts its behavior over *comparable* epochs, where the performance is compared using the definition in the CS step to compute a general ranking of systems or performance changes across common or different queries; and
3. detects *unexpected* behaviour of s_t at given epochs.

Mainly, the Longitudinal Analysis step has two parameters: the systems S_t on which we focus and the CS method.

$$LA = (S_t, CS_{method})$$

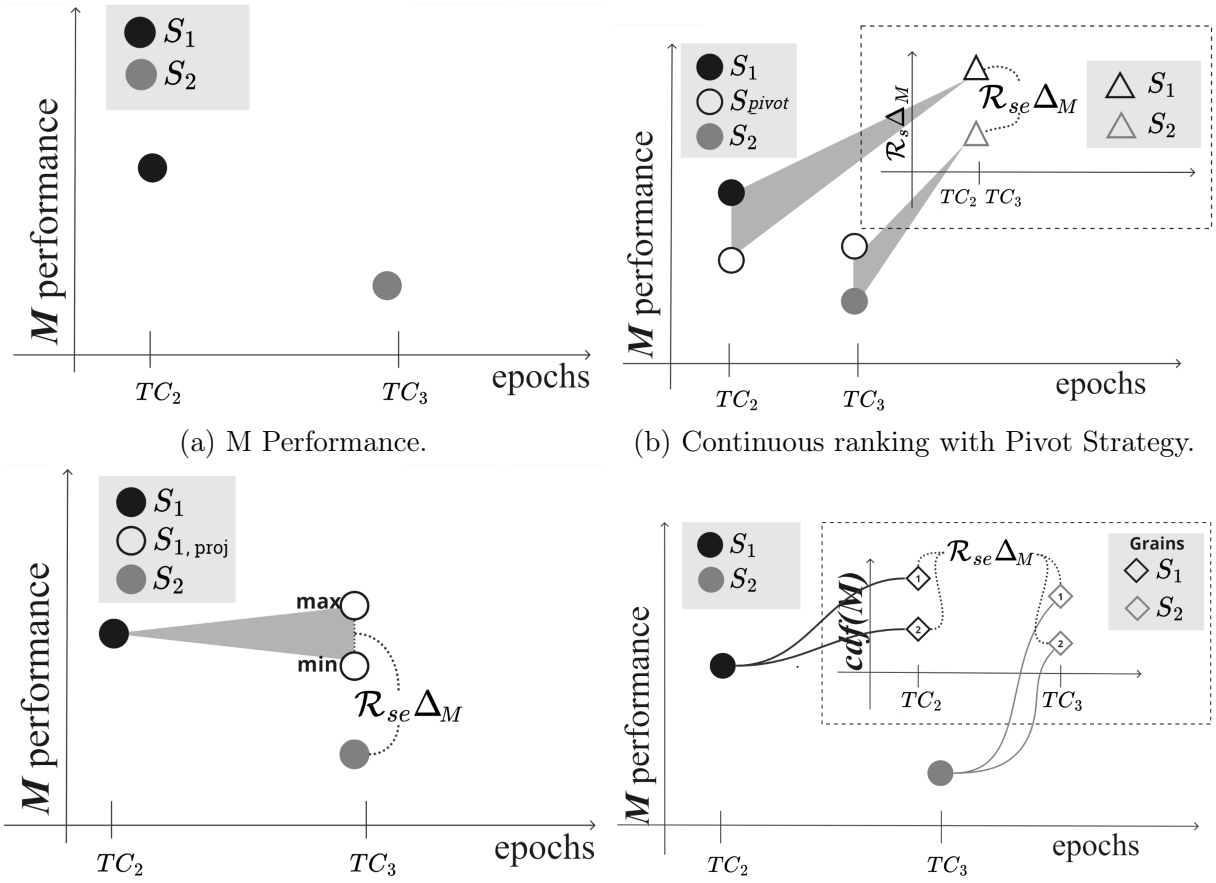
Depending on the method selected for the CS step, the longitudinal Analysis can be implemented as follows (Figure 4.7):

a) A Continuous-Ranking Analysis, if $CS_{method} = PIV$. In this analysis, the ranking is calculated at each epoch, validating the pivot selection using previous epochs.

b) An Expected Performance analysis, if $CS_{method} = PRO$. In this case, an expected performance is measured using the projection functions for each system evaluated in the previous epochs. Then the $\mathcal{R}_{se}\Delta_M$ is computed to compare the performance difference between systems between the performance of the current system versus the expected performance of past systems.

c) A Grain Analysis, if $CS_{method} = GRA$. Using grains we compare the performance of the systems, considering standardized values in a subset of queries. We analyze how the performance of a system change across epochs at each grain and compare the performance of two or several systems at each grain level.

Figure 4.7 illustrates how each method computes $\mathcal{R}_{se}\Delta_M$ for two systems evaluated in different epochs of an ETC (fig. 4.7a). The pivot strategy (fig. 4.7b) propose to evaluate the S_{pivot} at each epoch, then the $\mathcal{R}_{se}\Delta_M$ is computed in terms of $\mathcal{R}_s\Delta_M$ for each system (represented as triangles). Using the projection comparison (fig. 4.7c) we compute an expected performance ($S_{1,\text{proj}}$) composed by a maximum and minimum score, then $S_{1,\text{proj}}$ is compared to S_2 performance. Finally, using the grain comparison (fig. 4.7d) we compute $\mathcal{R}_{se}\Delta_M$ at each standardized performance grain (grains represented as diamonds).



(a) M Performance.

(b) Continuous ranking with Pivot Strategy.

(c) Expected Performance analysis with Projection Comparison.

(d) Grain Analysis using standardized performance scores at each grain.

Figure 4.7: Performance score transformation for the computation of $\mathcal{R}_{se}\Delta_M$.

4.4 Experiments

Our proposal is dedicated to handling evolving test collections. We present and evaluate our continuous evaluation framework using simulated evolving test collections described in Chapter 3. This allows us to evaluate our proposal on epochs for which we control the building process. As a real use case, we present our proposal using the LongEval evolving test collection.

4.4.1 Data

To evaluate our proposal, we use two simulated test collections, Robust$_e$, from TREC-Robust [131] and TREC-COVID$_e$, from TREC-COVID [127]. The simulation follows the same procedure described in section 3.5.1. As described previously, Robust$_e$ and TREC-COVID$_e$ are time-based ETCs composed of 41 epochs with 90% overlapped documents between successive epochs. We define 12 systems as the reference systems set S, which is composed of: BM25, PL2, Dirichlet language model (DLM) and TFIDF, in three versions: with Bo1 or KL pseudo-relevance feedback and without it. We define three test systems S_t as BM25, PL2 and TFIDF with RM3 pseudo-relevance feedback model. All systems are implemented with pyterrier using default parameters in the IR model and in the pseudo-relevance feedback when applied. The continuous evaluation framework is implemented using seven evaluation metrics (M): Average Precision (AP), normalized discount cumulative gain (ndcg), ndcg@10, reciprocal rank (recip_rank), binary preference (bpref), Precision@10 (P_10) and R-precision (Rprec).

4.4.2 Comparability Validation Step

The Continuous Evaluation Framework starts checking if successive epochs are similar enough to have a meaningful comparison of systems evaluated in different epochs of the evolving test collection (as described in section 4.3.3).

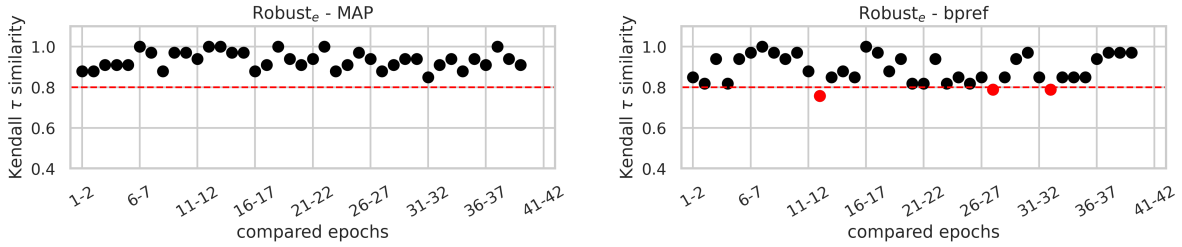
The parameters of the CV steps are S as the list of reference systems, M as the list of evaluation metrics and we define $\tau_{CS} = 0.8$, as in [94]:

$$CV = (S, M_i, 0.8), \forall M_i \in \{MAP, bpref\}$$

We report our results using MAP and bpref metrics as an example of evaluated metrics.

Figure 4.8 presents Kendall's τ values, representing the similarity between the rankings of 12 reference systems across successive epochs in the Robust*e* experiment, using the MAP (fig. 4.8a) and bpref (fig. 4.8b) metrics. With $\tau_{CS} = 0.8$, we find that all epochs are comparable (indicated by black dots) when evaluated using the MAP metric, and more than 90% of the epochs are comparable in the case of bpref. Similarly, TREC-COVID$_e$ epochs are comparable in a big percentage (in Figure 4.9). Using also $\tau_{CS} = 0.8$ for the

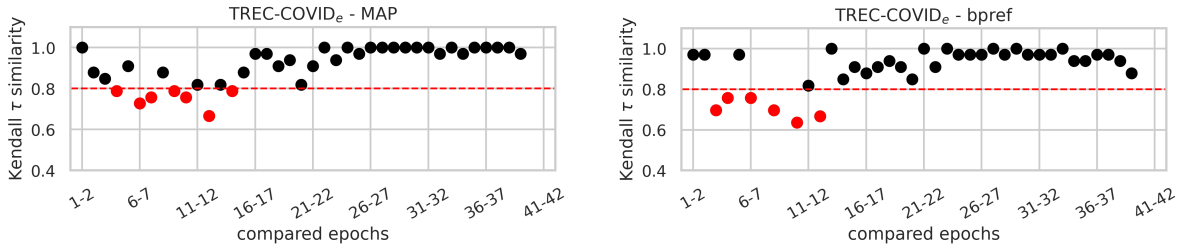
Robust$_e$ experiment, we observe that 80% of the epochs are comparable for both the MAP (fig. 4.9a) and bpref (fig. 4.9b) metrics. Non-comparable epochs (represented by red dots) are mainly concentrated in the initial epochs of the ETC. In contrast, the last epochs demonstrate a high degree of comparability, with a τ value of 1 observed in 11 epochs when evaluated using the MAP metric.



(a) MAP

(b) bpref

Figure 4.8: Comparability Validation - Robust$_e$.



(a) MAP.

(b) bpref.

Figure 4.9: Comparability Validation - TREC-COVID$_e$.

According to these results, in the next section, we define the comparison strategy for the comparable epochs in Robust$_e$ and TREC-COVID$_e$.

4.4.3 Comparison Strategy

In this section, we define the parameters for the three comparison strategies (see section 4.3.4). With the Pivot strategy, we select and validate the pivot system using the correctness evaluation. Using the Projection strategy, we compute the standardization and projection functions. Finally, we define and validate the comparison Grains for successive epochs of the ETCs.

4.4.3.1 Pivot Strategy

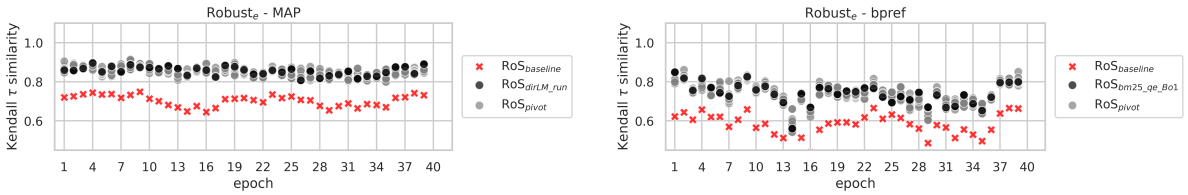
The Pivot Strategy (section 4.3.4.1) needs a baseline system as a Pivot to create an RoS evaluated in different epochs. The parameters of this step are the set of reference systems S used to select a Pivot and the evaluation metrics in M:

$$PIV = (S, M_i), \forall M_i \in \{MAP, bpref\}$$

In the following, we define a Pivot using the reference systems as candidates and select the one that generates the most *correct* RoS at each compared epoch. The selection process involves computing the correctness of the RoS (RoS_{pivot}) built using the Pivot. The correctness is measured by Kendall's τ similarity between the RoS_{pivot} and the ground truth $RoS_{reference}$. To validate the selection, we compare the correctness of the RoS_{pivot} with that of a baseline RoS_{baseline}, as described in Section 4.3.4.1.

Figure 4.10 presents the correctness evaluation of the candidate systems in Robust$_e$ for MAP (fig. 4.10a) and bpref (fig. 4.10b). The correctness of each pivot system is represented as dots, while the baseline correctness is shown as a red cross. We highlight one candidate system (dirLM in MAP and bm25 with Bo1 in bpref) in black, which has been selected as the Pivot due to higher correctness in all the epochs. Although no pivot consistently exhibits the highest correctness across all epochs, the correctness of all pivots is larger than the correctness of the baseline for all epochs. With the MAP metric, all pivots' correctness values exceed 0.8, while with bpref, the correctness ranges from 0.6 to 0.9.

The TREC-COVID$_e$ results (Figure 4.11) align with those of Robust$_e$. The correctness of all pivots surpasses the baseline correctness. In this case, PL2-KL presented the best results for MAP (fig. 4.11a) and bpref (fig. 4.11b). The correctness results are similar for both metrics, with values close to or exceeding 0.8 for most of the comparable epochs. Consistent with the comparability results, the best correctness values are observed in the final epochs of the ETC.

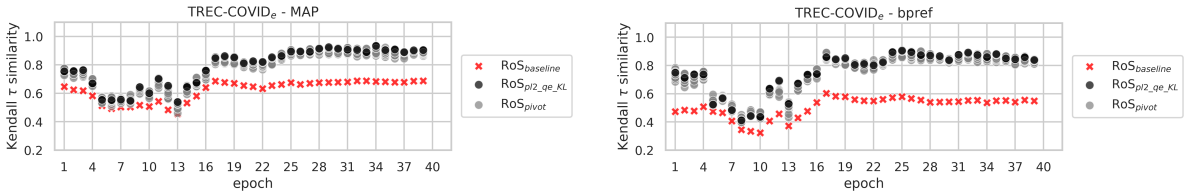


(a) MAP.

(b) bpref.

Figure 4.10: Pivot Selection - Robust$_e$.

These results demonstrate that the Pivot strategy generates a more correct ranking of systems compared to the baseline approach in an ETC. By evaluating the correctness



(a) MAP.

(b) bpref.

Figure 4.11: Pivot Selection - TREC-COVID$_e$.

of the Pivot, we ensure the selection of a reliable system to build an RoS that compares systems across different epochs.

4.4.3.2 Projection Comparison

To compare the performance of systems across test collections, we now experiment with a projection comparison using the standardization and projection functions. These functions are computed using the set of reference systems S for all the evaluation metrics in M. In this section, we present the score standardization methods applied to Robust$_e$ and TREC-COVID$_e$.

The standardization functions transform the raw values through a cumulative distribution function (cdf). The parameters of the distribution are computed from the performance values of the reference systems. We decided to use the Uniform distribution (U-cdf) as it has been shown to work well with a small number of systems and is more consistent than the normal distribution for comparing pairs of systems across different test collections [99]. Additionally, due to the limited number of reference systems (12 in our case versus the 110 systems used in the work proposal [123]), empirical standardization functions were not feasible.

The Projection parameters are defined as follows:

$$PRO = (S, M_i, \text{U-cdf}), \forall M_i \in \{MAP, bpref\}$$

To illustrate the transformation from raw values to standardized and projected scores, we present these functions computed in two epochs (TC_2 and TC_3) for a specific query in Robust$_e$ and TREC-COVID$_e$. For *Robust$_e$* w present the transformations of $Q_3 02$ (Figure 4.12). From left to right in Figure 4.12, the first graphic shows the U-cdf(MAP) standardization function for TC_2, which is computed according to our reference systems (denoted in red in the graphic). The second graphic shows the same for TC_3. Using previous standardization functions, we compute the projection function (in the third graphic) that transforms the performance score from TC_2 to TC_3. It is important to

notice that the standardization functions are not bijective: a performance of 0 to 0.35 is transformed to the same value ($U - cdf(MAP)(0) = U - cdf(MAP)(0.35) = 0$) in both epochs. As described in section 4.3.4.2, we approximate the projection function with a minimum and maximum score, which is why the third graph displays two functions for the projection (ranging from 0 to 0.3). A similar situation occurs with the bpref metric and with the TREC-COVID$_e$ example (in Figure 4.13).

Another result from this step is the variability of the standardization functions when computed across different epochs. This can be observed in TREC-COVID$_e$ (in Figure 4.13) for MAP metric. We can see that the difficulty changes, as a MAP score of 0.6 in TC_2 is transformed to a MAP performance of 0.5 in TC_3, indicating that query Q_{10} is harder in TC_3 compared to TC_2, according to the set of reference systems. This difficulty difference across the epochs supports the need for projection functions to compare the system's performance in an epoch of interest.

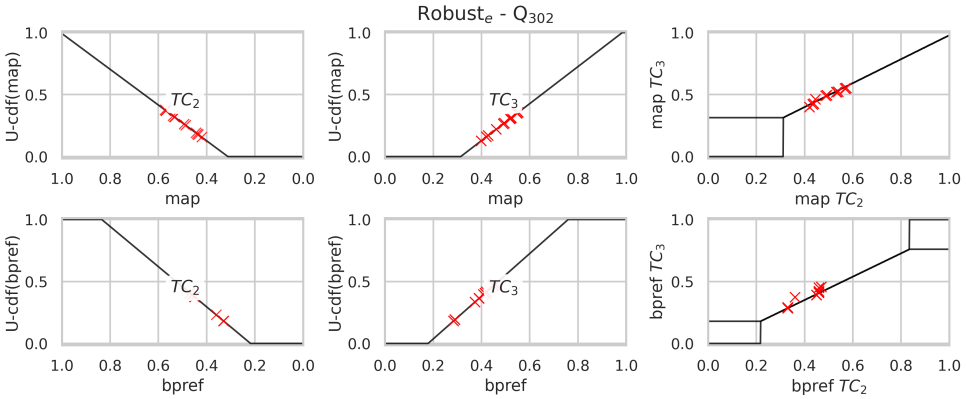


Figure 4.12: Standardization Functions - Q_{302} Robust$_e$.

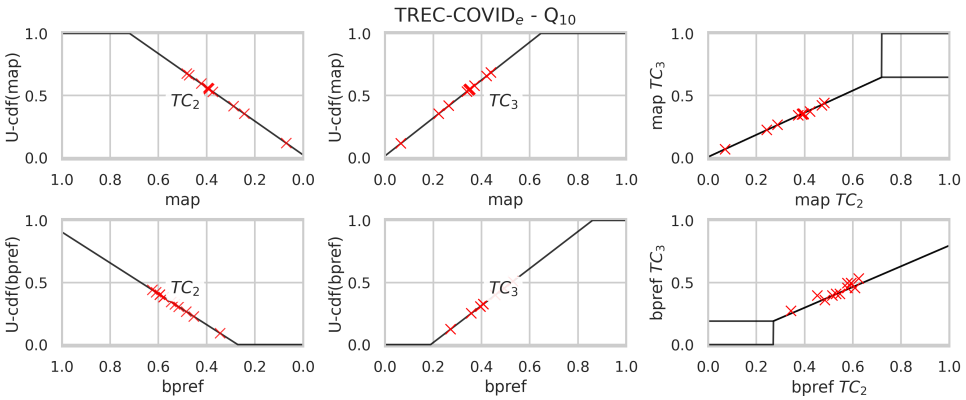


Figure 4.13: Standardization Functions - Q_{10} TREC-COVID$_e$.

From the Robust$_e$ and TREC-COVID$_e$ standardization and projection examples, we observe that no function is bijective, leading to the computation of the projection as two functions representing the minimum and maximum projected scores, as described in section 4.3.4.2. Therefore, the projection is defined as a range of minimum and maximum projected scores.

4.4.3.3 Grain Comparison

The grain comparison proposes to compare the performance of systems using comparable, yet different, query sets across the epochs. This step is defined by the set of reference systems S, and evaluation metrics M, using utilizing the U-cdf standardization functions, as in the PRO step, and we define a comparison threshold τ_{GRA} set to 0.7 as we still need to ensure a high level of similarity between RoS using the grains, but considering that the epochs were already tested as comparable in the CV step:

$$GRA = (S, M_i, \text{U-cdf}, 0.7), \forall M_i \in \{MAP, bpref\}$$

To show interest in using grains, we explore two granularities based on query groups. The first, denoted as G_1, considers all queries belong to a single category with a performance value greater than 0. The second, G_2, builds three categories of queries. More precisely, G_2 is built using the following process:

1. we define three performance intervals
   - $I_{high} = [0.65, 1]$,
   - $I_{medium} =]0.35, 0.65[$,
   - $I_{low} = [0, 0.35]$;
2. a query is assigned to a category if at least 40% of the system's standardized performance values for that query fall within the corresponding interval.

To decide which epochs are comparable using each grain, we validate the comparability of the epochs in the ETC by considering only the queries that belong to the grain. In other words, we assess the RoS similarity between epochs based on the performance of the queries within the grain category.

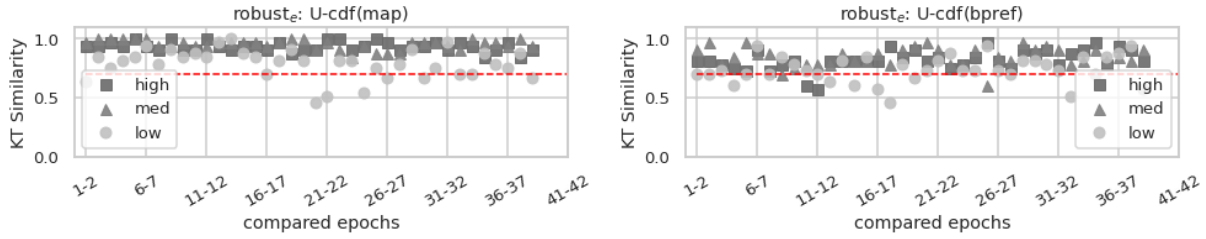
Figure 4.14 presents the comparability between epochs in Robust$_e$ using the grains defined in G_2 with U-cdf(MAP) and U-cdf(bpref). Each grain category is represented by different symbols: square for high, triangle for medium, and circle for low. In general, the comparability of the high and medium categories remains consistently high across all epochs ($\tau > 0.7$ in almost all epochs). Only two epochs are not comparable when using high grains, and only one epoch is not comparable when using medium grains, both with U-cdf(bpref). However, there is a lower level of comparability observed for the low

category in some epochs. Six epochs are not comparable when using low grains with U-cdf(MAP), and seven epochs are not comparable when using low grains with U-cdf(bpref). This lower comparability may be related to the number of queries considered in each grain category, with a mean of 63, 117, and 40 queries in the high, medium, and low categories, respectively.

In TREC-COVID$_e$ (Figure 4.15), the comparability of epochs using U-cdf(MAP) (fig. 4.15a) shows that ten epochs are not comparable in the high category, all epochs are comparable in the medium category, and seven epochs are not comparable in the low category. When using U-cdf(bpref) grains (fig. 4.15b), the comparability of epochs is similar for the medium and low categories, with all epochs being comparable. However, the high category is not comparable in 15 epochs, indicating a larger discrepancy in performance among systems for those epochs.

This indicates a lower level of comparability in TREC-COVID$_e$ compared to Robust$_e$. This may be attributed to the smaller number of queries in the dataset, only 50 queries. The high and low categories have a mean of only six queries, while the medium category has a mean of 27 queries. This limited number of queries in the high and low categories may contribute to the decreased comparability observed.

In both Robust$_e$ and TREC-COVIDe, the comparability of G_1, where all queries are in one category, is larger than the threshold τ_{GRA} in all epochs. This is consistent with Figure 4.8 and Figure 4.9, as, in fact, the main difference is that we are looking at standardized values instead of non-standardized ones.



(a) U-cdf(MAP).

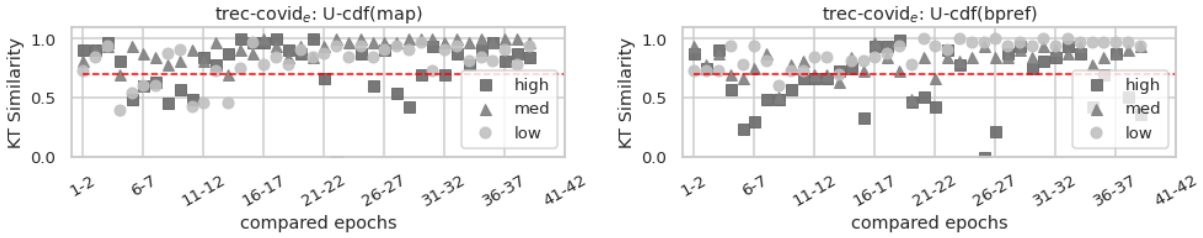
(b) U-cdf(bpref).

Figure 4.14: Grain epochs Comparison - ROBUST$_e$.

This shows that by using adequate categories, we are able to locate specific sets of queries while remaining comparable. This result supports a detailed analysis of performance variations of the tested systems in an ETC.

4.4.4 Longitudinal Analysis

In this section, we present the results of our continuous evaluation framework. We show how to compute the $\mathcal{R}_{se}\Delta_M$ between systems evaluated in different epochs of the ETC,



(a) U-cdf(MAP). (b) U-cdf(bpref).

Figure 4.15: Grain epochs Comparison - TREC-COVID$_e$.

according to each Comparison Strategy. First, we analyze the ranking of systems across the ETC (LA_1); second, we compare the performance using projections (LA_2); and finally, we compare standardized performance at comparable grains (LA_1). The Longitudinal Analysis parameters are:

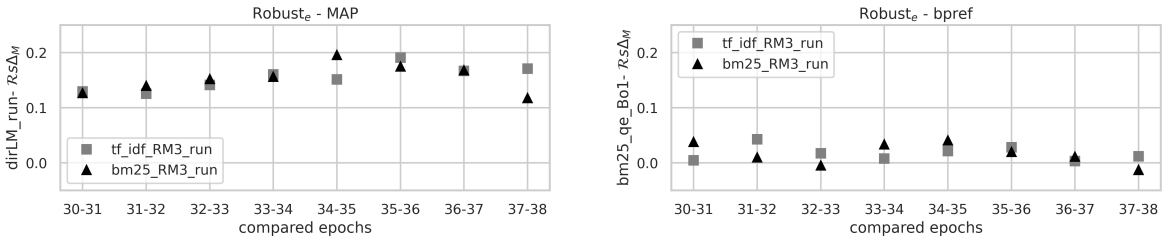
$$LA_1 = (S_t, PIV); LA_2 = (S_t, PRO); LA_3 = (S_t, GRA)$$

4.4.4.1 Continuous Ranking of Systems

The first Longitudinal Analysis (referred to as LA_1) is a continuous ranking of systems built using the selected pivot in the PIV step. The goal of a Pivot Ranking is to create an overall ranking considering two comparable epochs, where different systems are evaluated in each epoch. To illustrate this evaluation scenario, we consider a first system st_1 (tf_idf_RM3) evaluated only in the first epoch and a second system st_2 (bm25_RM3) evaluated only in the second epoch. This means that we only have information about the performance of st_1 evaluated in TC_1 as $M(TC_1, st_1)$ and st_2 evaluated in TC_2 as $M(TC_2, st_2)$.

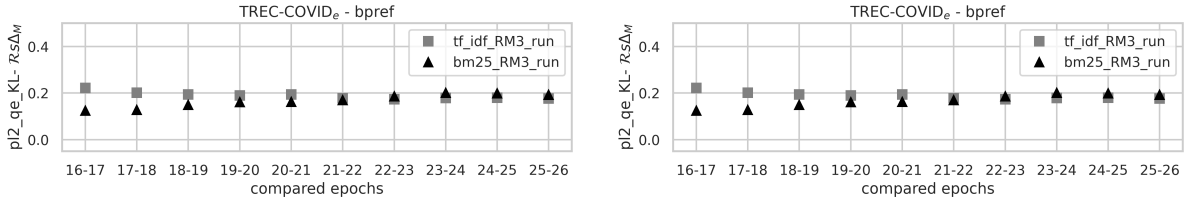
Figure 4.16 displays the continuous ranking for systems st_1 and st_2 across eight pairs of compared epochs of Robust$_e$ using MAP and bpref. Comparing the performance of these systems using a Pivot, we observe that their rankings vary across the compared epochs. The improvement of st_2 to the Pivot defines that st_2 outperforms st_1 in three epochs when evaluated with MAP (fig. 4.16a) and four times in bpref (fig. 4.16b). In TREC-COVID$_e$ (Figure 4.17), we observe a similar trend where the rankings of systems st_1 and st_2 change across ten pairs of epochs. Comparing the $\mathcal{R}_s\Delta_M$ values, system st_1 outperforms st_2 from epochs 16-17 to 21-22, then st_2 has higher $\mathcal{R}_s\Delta_M$ values from epochs 22-23 until epochs 22-26 for MAP (fig. 4.17a) and bpref (fig. 4.17b).

The Pivot Comparison Strategy is able to rank systems from different epochs while keeping the Pivot system constant. The pivot becomes particularly useful when the performance of a system is unknown in certain epochs, and we want to compare the



(a) MAP. (b) bpref.

Figure 4.16: Continuous Ranking of systems. Robust$_e$.



(a) MAP. (b) bpref.

Figure 4.17: Continuous Ranking of systems in TREC-COVID$_e$.

systems. To demonstrate the value of continuous ranking using a Pivot system, we assess the ranking agreement between pairs of test systems at each compared epoch against a ground truth.

To establish ranking agreement, we compare the order of systems st_1 and st_2 using the pivot strategy with the ground truth. By employing simulated ETCs, we can compute the true ranking between two systems in a reference epoch, denoted as $TC_{ref(i,i+1)}$, which is built as the union of TC_i and TC_{i+1}. We then calculate the proportion of agreement between the Pivot ranking of st_1 and st_2 in the compared epochs TC_i and TC_{i+1} and the real ranking of st_1 and st_2 in $TC_{ref(i,i+1)}$.

Table 4.2 shows the number of times that the order between systems st_1 and st_2 agrees with the ground truth at long of Robust$_e$ and TREC-COVID$_e$ epoch pairs. We compare the results to a baseline that orders the systems according to their performance values at each epoch. In Robust$_e$, the pivot orders the test systems agreeing with the ground truth in 86% of the cases with MAP, whereas the baseline ranking achieves agreement in 83% of cases. For the bpref metric, the pivot ranking agreement is 89%, while the baseline ranking achieves 82% agreement. Moving to the TREC-COVID$_e$ dataset, we observe higher agreement values for the pivot rankings compared to Robust$_e$. The pivot ranking achieves 95% agreement with the ground truth for the MAP metric, surpassing the 88%

agreement achieved by the baseline ranking. Similarly, for the bpref metric, the pivot ranking achieves 94% agreement, while the baseline ranking achieves 79% agreement. In conclusion, the Pivot ranking consistently outperforms the baseline ranking in terms of agreement with the ground truth.

Robust$_e$			
Metric	Pivot System	Pivot	Baseline
MAP	dirLM	0.86 ± 0.19	0.83 ± 0.19
bpref	bm25 Bo1	0.89 ± 0.17	0.82 ± 0.19
TREC-COVID$_e$			
Metric	Pivot System	Pivot	Baseline
MAP	pl2 KL	0.95 ± 0.05	0.88 ± 0.06
bpref	pl2 KL	0.94 ± 0.05	0.79 ± 0.10

Table 4.2: Pivot and baseline agreement with the ground truth (mean ± standard deviation).

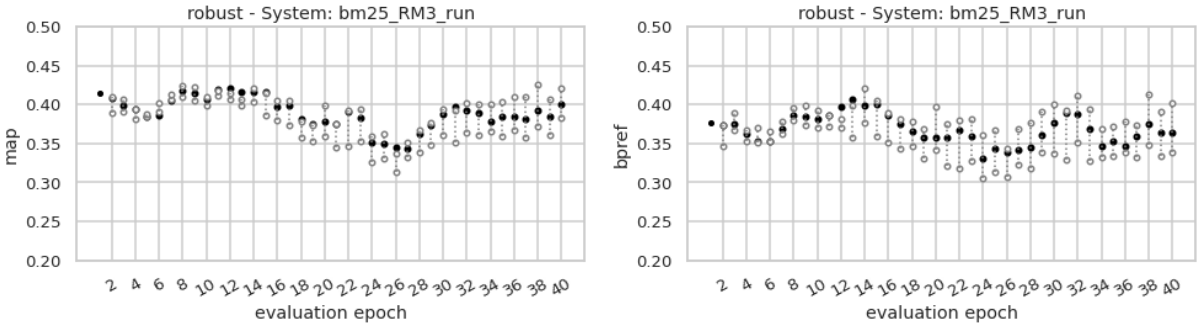
The Pivot is able to define which system outperforms the other when they are compared in two different test collections agreeing with the real results more than ranking them according to their absolute values (baseline). The $\mathcal{R}_{se}\Delta_M$ computed with the Pivot system is defined in terms of the improvement of the tested system versus the Pivot and not in the scale of the performance metric.

4.4.4.2 Expected Performance Analysis

The second Longitudinal Analysis (defined as LA_2) is an expected performance analysis using standardization functions to project the performance of one test system st_1 evaluated in one epoch (TC_1) to another epoch (TC_2). The purpose of this analysis is to compare systems based on a specific epoch of interest. By projecting the expected performance of st_1 in TC_2, we can compare it to any other system evaluated in TC_2 This approach enables us to understand how changes across epochs impact the performance of the tested systems.

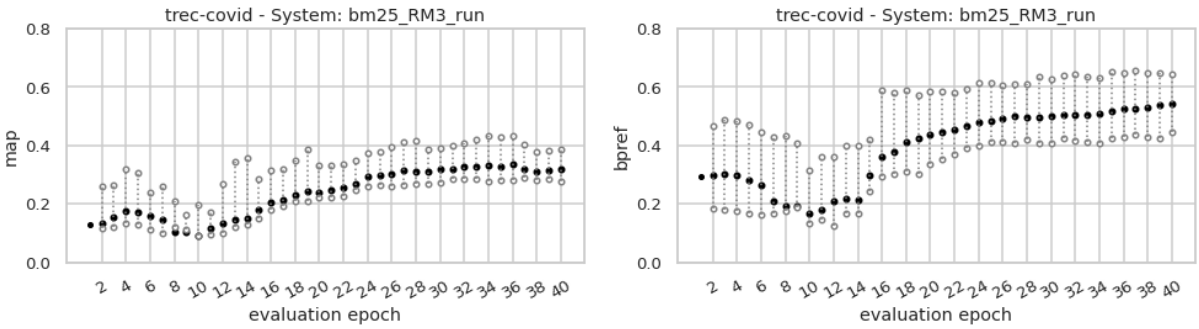
Figure 4.18 presents the results for Robust$_e$, comparing the real performance of the tested system st_1 (represented by black dots) with its expected performance in the previous epoch (i.e., the expected performance in epoch TC_2 corresponds to the projection of st_1's performance in epoch TC_1). In terms of MAP (fig. 4.18a), the performance of st_1 aligns with the expected performance in the majority of epochs, as the real performance falls within the expected performance range 75% of the time. For the remaining

25% of cases, the expected performance is very close to the real performance. As for bpref (fig. 4.18b), the real performance consistently falls within the expected performance range 95% of the time. The same trend can be observed in TREC-COVID$_e$, where the MAP (fig. 4.19a) real performance is outside the expected performance range only twice. For bpref metric (fig. 4.19b), the real performance always falls within the expected range. However, compared to Robust$_e$, the expected performance ranges in TREC-COVID$_e$ are larger.



(a) MAP. (b) bpref.

Figure 4.18: Expected performance analysis - Robust$_e$.



(a) MAP. (b) bpref.

Figure 4.19: Expected performance analysis - TREC-COVID$_e$.

$\mathcal{R}_{se}\Delta_M$ of two systems (st_1 evaluated in TC_1 and st_2 in TC_2) is computed as the difference between st_2's real performance. $M(TC_2, st_2)$, and st_1's expected performance, $M_{proj}(TC_2, TC_1, st_1)$ (see section 4.3.5). As the expected performance is a range of values, we evaluate which value in the range is a good predictor of performance change.

Considering ER as the expected range of performance, we evaluate four scenarios to calculate the performance change in two consecutive epochs ($epoch_i$, $epoch_j$):

- Comparing real performance value at $epoch_i$ versus the mean value of ER, denoted as mean(ER), at $epoch_j$,
- Comparing real performance value at $epoch_i$ versus the minimum value of the ER, denoted as min(ER), at $epoch_j$,
- Comparing real performance value at $epoch_i$ versus the maximum value of the ER, denoted as max(ER), at $epoch_j$,
- Comparing the mean value of ER at $epoch_i$ versus the mean value of the ER at $epoch_j$.

By assessing these scenarios, we determine the number of times that the expected performance correctly indicates an improvement when the real performance actually improves, and the number of times that the expected performance indicates a decrease when the real performance does decrease. This analysis helps us identify which projected value in the range is a reliable predictor of performance change, which is then used to compute $\mathcal{R}_{se}\Delta_M$ for the systems.

Table 4.3 compares the four scenarios in Robust$_e$ and TREC-COVID$_e$. We report the mean proportion of agreement between the expected change and the real change across all test systems and all the successive epochs. In the case of Robust$_e$, both MAP and bpref metrics show that the best indicator of a performance change is when comparing the real performance and the mean of the expected performance range. In TREC-COVID$_e$, MAP results in a better prediction of an improvement when comparing both expected performance means. In the case of bpref, the results are like Robust$_e$ with the mean expected performance with the best results. Although the overall agreement proportions are slightly lower in TREC-COVID$_e$ compared to Robust$_e$, the trend remains consistent.

		Robust$_e$		**TREC-COVID$_e$**	
$epoch_i$	$epoch_j$	MAP	bpref	MAP	bpref
real	mean(ER)	0.85	0.75	0.68	0.75
real	min(ER)	0.66	0.56	0.34	0.29
real	max(ER)	0.72	0.65	0.67	0.75
mean(ER)	mean(ER)	0.72	0.70	0.75	0.69

Table 4.3: Agreement of expected performance change.

By considering the appropriate projected value, we can compute $\mathcal{R}_{se}\Delta_M$ and better understand the performance changes of the evaluated systems across different epochs.

Given the results of Table 4.3, we propose to use the mean of the expected range in the $\mathcal{R}_{se}\Delta_M$ computation.

The expected performance analysis is useful to compare one system and the next version of itself from the perspective of the current test collection. To understand how the changes in the test collection could affect the performance of the system. As it is limited by the common queries in consecutive epochs, we propose a Grain analysis that compares performances in meaningful splits of the test collections.

4.4.4.3 Grain Analysis

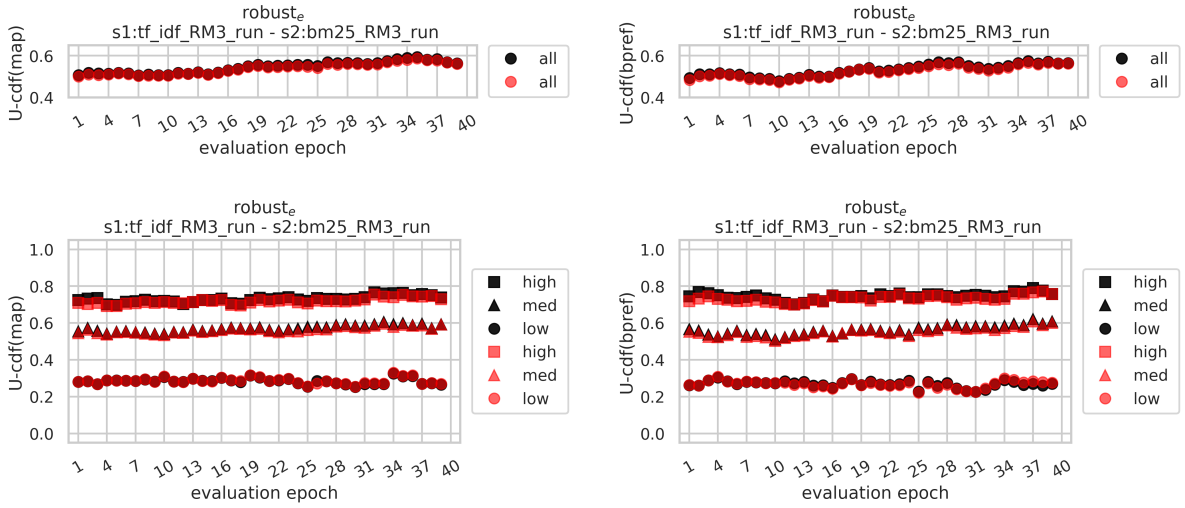
In the third Longitudinal Analysis (LA_3), we employ a Grain analysis using the predefined granularities from the GRA step. This analysis allows us to compare the performance of test systems using standardized performance values at each grain. We compare tf_idf_RM3 (later represented in black) and bm25_RM3 (in red).

For Robust$_e$, the results of the Grain analysis are presented in Figure 4.20, which compares TF_IDF_RM3 (in black) and BM25_RM3 (in red). In the top row, we observe that the G_1 granularity does not exhibit any specific behavior for either test system. The performance values remain relatively stable within the range of $0.5 < U - cdf_M < 0.58$ for both metrics and systems. For the G_2 granularity (bottom row of Figure 4.20), we see that the systems perform as expected in the low, medium, and high-performance queries, demonstrating consistency and alignment with the performance categories across the granularities.

In the case of TREC-COVID$_e$, the performance at G_1 is very stable (Figure 4.21, first row). However, a notable difference compared to Robust$_e$ is observed when considering the G_2 granularities. In this case, the performance of the test systems becomes highly unstable for high and low-performance queries.

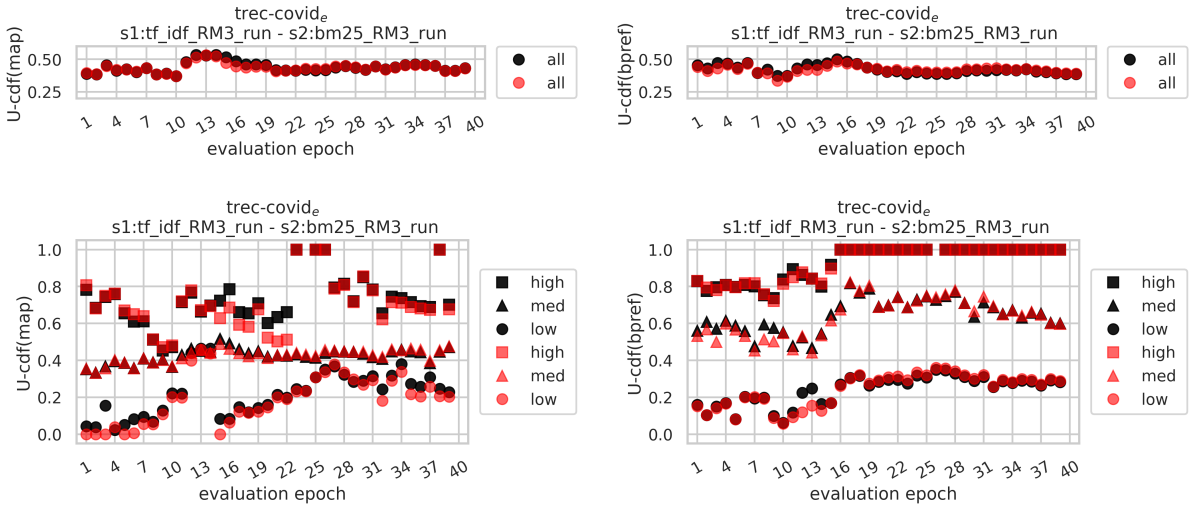
By utilizing the Grain analysis, we can further analyze the specific performance characteristics of the test systems. For example, in TREC-COVID$_e$, the high-performing queries in epoch 20 (comparable) have a U-cdf(MAP) performance lower than expected. We can conclude that the test system changes its capacity to process high-performance queries dropping to 0.5 its U-cdf(MAP) performance. Additionally, in the comparable epochs 17, 18, and 19, the medium-performing queries have a better performance than expected, with U-cdf(bpref) > 0.75, surpassing the medium range. Through the Grain analysis, we can pinpoint specific performance trends and deviations within different granularities, providing valuable insights into how the test system's performance varies across query grains and epochs.

We have presented and validated our continuous evaluation framework using the two simulated evolving test collections (presented in section 3.3). At each comparison analysis, we have validated our findings and the application of each strategy, which allows the



(a) MAP. (b) bpref.

Figure 4.20: Grain Comparison - Robust$_e$.



(a) MAP. (b) bpref.

Figure 4.21: Grain Comparison - TREC-COVID$_e$.

computation of $\mathcal{R}_{se}\Delta_M$ across systems evaluated in different test collections of an ETC.

4.4.5 LongEval Use Case

We apply our Evaluation Framework to LongEval ETC. As a reminder, LongEval is composed of three test collections at different epochs: epoch one corresponds to June, the second to July, and the third epoch corresponds to September. The lag between epochs is measured in months. Between June and July, there is a lag of one. Between July and September, the lag is two. For completeness, we also report the comparison with a lag distance of three, comparing June and September.

As reference systems, we use 12 IR implementations evaluated using LongEval epochs with seven performance metrics (IR systems and metrics described in experiments, section 4.4).

Step 1. Comparability Validation Step: Table 4.4 presents the comparability between epochs with a lag of one, two, and three. The comparability is measured as Kendall's τ similarity between the ranking of the reference systems. MAP, nDCG, and P@10 are highly comparable in all three lag distances with $\tau > 0.7$ (marked in gray in table 4.4). For the rest of the metrics, we only consider a valid evaluation when the comparability is larger than 0.65 (marked in light gray in Table 4.4), limiting the comparison of bpref to June-July and June-September. for the case of R-Precision, we only consider June-September; for reciprocal rank (RR) we compare the June-July and June-September; and finally, nDCG@10 is used on June-July and June-September.

These results align with our findings from the analysis of simulated evolving test collections. In both cases, we observed highly comparable epochs as well as epochs that were not comparable based on the chosen metrics. Therefore, it is crucial to consider the comparability between epochs when conducting a continuous evaluation and selecting appropriate metrics for comparison.

Since nDCG and P@10 demonstrate high comparability across all three lag distances, these metrics are suitable for demonstrating the continuous evaluation framework using LongEval.

Step 2. Pivot Strategy Comparison: As a comparison strategy, we first implement the Pivot strategy. We assess the ability of the reference systems to create a correct RoS for the purpose of pivot selection.

Figure 4.22 presents the correctness of the RoS_{pivot} versus a $RoS_{baseline}$ for nDCG and P@10 metrics at each epoch of LongEval. Similar to our findings with simulated ETCs, it is not possible to find one best Pivot across all the compared epochs. However, all candidate pivots demonstrate the ability to construct more correct RoS compared to the baseline.

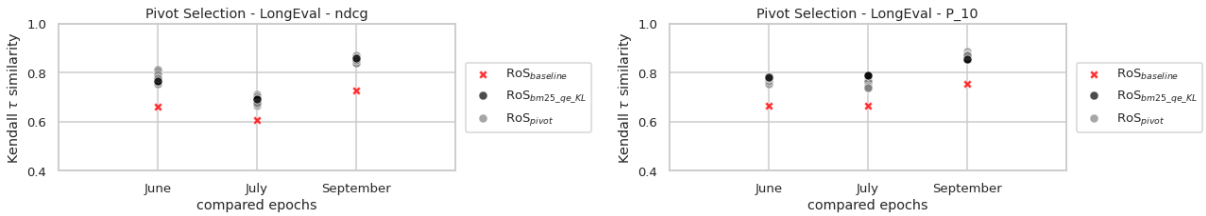
Considering the results with the simulated ETCs, the mean correctness achieved with the baseline across epochs is 0.65 in Robust_e and 0.6 in TREC-COVID_e (with all metrics), while the mean correctness of a pivot is 0.7 and 0.8, respectively. A similar trend is found

Metric	Kendall's τ(S,TC_i,lag) June - July lag=1	July - Sept lag=2	June - Sept lag=3
MAP	0.746	0.806	0.777
bpref	0.674	0.441	0.723
nDCG	0.959	0.845	0.88
R-Precision	0.296	0.426	0.664
RR	0.851	0.430	0.670
ndcg@10	0.769	0.531	0.645
P@10	0.889	0.860	0.850

Table 4.4: RoS Kendall's τ similarities between epochs.

with LongEval. In this case, the mean correctness of the baseline is 0.7, and the mean correctness of the pivot is 0.8. Therefore, incorporating a pivot in the ranking process is a valid strategy, whether in a simulated ETC or in real scenarios like LongEval.

Finally, we choose BM25 with KL relevance feedback as the Pivot.



(a) nDCG.

(b) P@10.

Figure 4.22: Pivot Candidates versus Baseline in LongEval.

Step 2. Projection Comparison: In this step, we compute the standardization function for each query at each epoch with the reference systems. As the number of reference systems is only 12, we use U-cdf to standardize and project the performance values from one TC epoch to another one.

In Figure 4.23, we present an example of the standardization function for one query that is common to the July and September epochs. The first two graphs in the figure depict the standardization functions employed to compute the projection function for this particular query. As both standardization functions are not bijections, the projection considers a minimum and maximum range in both metrics. For example, any nDCG value from 0 to 0.6 in TC_{july} is transformed to the performance range 0-0.48 in TC_{sept}.

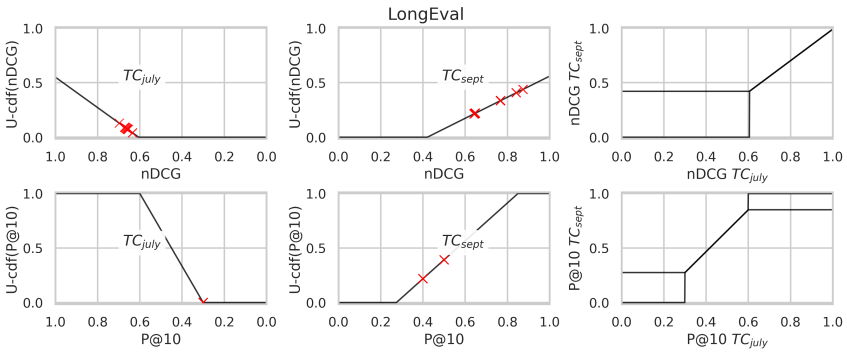


Figure 4.23: LongEval Query “Veal blanquette" with qid q07223839 in July, and qid q092210537 in September.

The projection functions are based on the standardization ones and are computed on the common queries in June, July, and September.

Step 2. Grain Definition: The grain definition compares the performance of systems at defined grains using standardized performance. We use the same granularities defined in section 4.4.3.3:

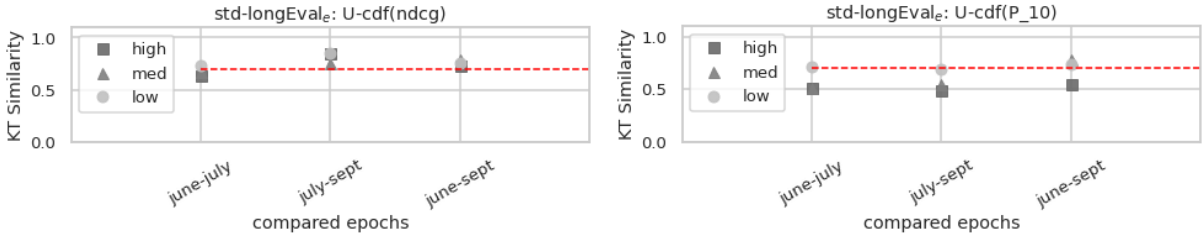
- $I_{high} = [0.65, 1]$,
- $I_{medium} =]0.35, 0.65[$,
- $I_{low} = [0, 0.35]$;

we define that a query belongs to the range if the performance of 40% of the reference systems is in the range.

In Figure 4.24, the comparability of each grain is assessed using Kendall's τ across epochs. The results show that the high, medium, and low grains are comparable for nDCG, indicating consistent performance patterns across epochs within each grain. However, for the P@10 metric, the comparability is limited to the low and medium grains between the June and September epochs.

Step 3. Continuous Ranking of Systems: We compare the Pivot comparison and the real performance of the systems. According to the pivot selection, we use BM25 KL as the pivot system.

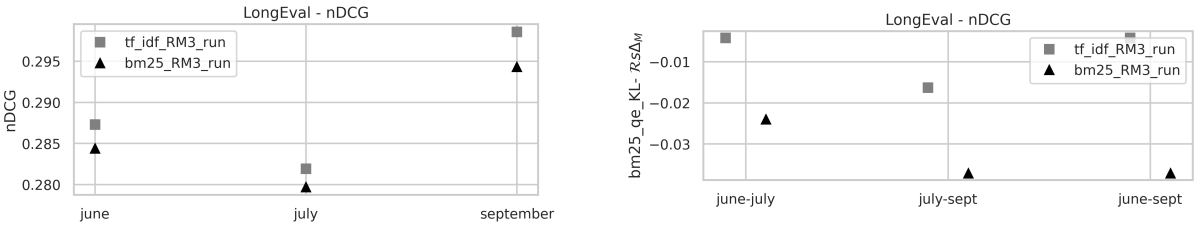
Figure 4.25 shows nDCG comparison of two systems in three moments. Figure 4.25a presents the real performance of both systems at each epoch. Figure 4.25b presents the continuous ranking, taking one system from the first epoch and comparing it to a second system evaluated in the next epoch. The continuous ranking defines that TF_IDF_RM3



(a) nDCG. (b) P@10.

Figure 4.24: Grain Comparability.

outperforms BM25_RM3 in all three compared epochs. P@10 results show the same results (Figure 4.26), TF_IDF_RM3 outperforms BM25_RM3 in all epochs.



(a) nDCG performance. (b) Pivot Continuous Ranking.

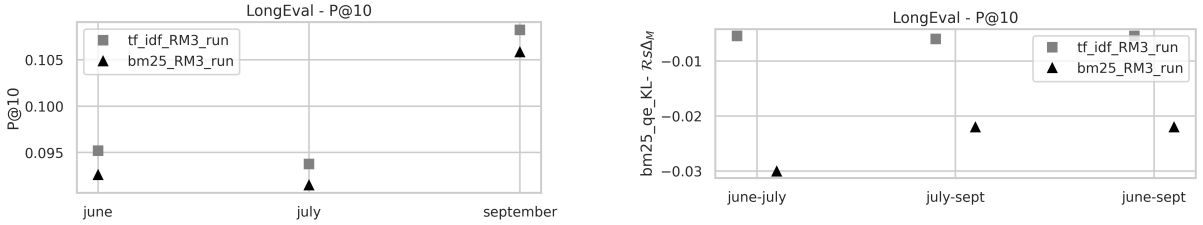
Figure 4.25: Performance by epoch and Continuous Ranking - nDCG.

According to the pivot, the difference between the systems in June-September is the largest difference measured with nDCG, and June-July when measured with P@10.

Step 3. Expected Performance Analysis: We compare the performance of the test systems using a subset of common queries that are shared across the epochs. This subset consists of 178 queries, with an average of 16 assessments per query. It is important to note that when evaluating performance on a smaller set of queries, the results may differ from the overall performance reported in Figure 4.25a and Figure 4.26a.

Figure 4.27 presents the real performance of the test systems (shown in black) along with the expected performance range for each test system, projected from the previous epoch. The results are presented for both nDCG and P@10 metrics.

In the case of nDCG (fig. 4.27a), the real performance of both systems in July falls within the expected performance range, indicating that the expected range is a good indicator of the performance change. However, in September, the real performance of both systems is lower than expected, suggesting a decline in performance compared to

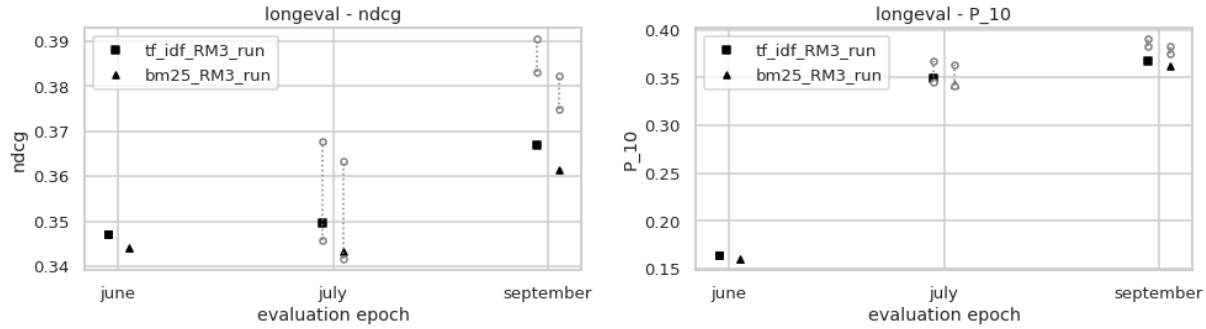


(a) P@10 Performance. (b) Pivot Continuous Ranking.

Figure 4.26: Performance by epoch and Continuous Ranking - P@10.

the projected values. It is worth noting that the difference between the real performance and the expected range is proportional to the difference between the two systems, with TF_IDF_RM3 consistently outperforming BM25_RM3. Similar patterns can be observed for the P@10 metric (fig. 4.27b), where the real performance in July aligns with the expected range, but in September, the performance is lower than expected for both systems. In conclusion, the performance increase in July and September was expected for both systems. It was also expected that TF_IDF_RM3 would have a better performance than BM25_RM3.

Compared to the results in the simulated ETCs, LongEval experiments also support the comparison of systems performance by the mean expected values to compute accurate $\mathcal{R}_{se}\Delta_M$ between systems.



(a) nDCG. (b) P@10.

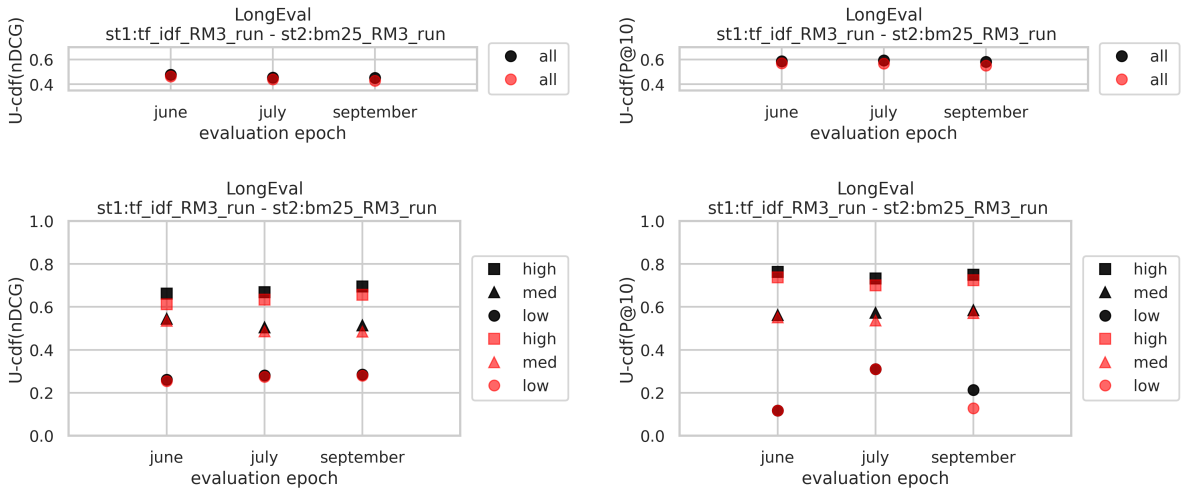
Figure 4.27: Expected Performance.

Step 3. Grain Analysis: In Figure 4.28, we present the grain comparison of test systems TF_IDF_RM3 (in black) and BM25_RM3 (in red). Figure 4.28a shows the results for U-cdf(nDCG). In the top graphic, we observe the G_1 grain, which considers all the

queries of each epoch. U-cdf(nDCG) remains stable across epochs, with TF_IDF_RM3 consistently outperforming BM25_RM3 in all three epochs.

In the bottom part of Figure 4.28, we present the comparison of systems at each granularity of G_2. Here, we analyze the performance of the systems at different performance levels: low, medium, and high. In this case, we detect that: for the low-performance queries, TF_IDF_RM3 demonstrates similar performance across epochs; For the mid-performance queries, TF_IDF_RM3 performs better in June compared to the other epochs; and for the high-performance queries, the performance of TF_IDF_RM3 improves with each epoch. This analysis highlights the importance of considering grains, as the standardized scores indicate a performance decrease that is opposite to the reality (shown in fig. 4.25a), where the real performance of the system is improving across the epochs. The findings are noticeable when using G_2 but not when considering the aggregated view of G_1, which can lead to incorrect conclusions in the continuous evaluation of systems.

Furthermore, Figure 4.28b presents the results for P@10, showing a similar trend to nDCG. TF_IDF_RM3 consistently outperforms BM25_RM3, and the comparable grains of G_2 (medium and low for the June-September comparison) also demonstrate an improvement in system performance. In contrast, the G_1 grain suggests a drop in standardized scores.



(a) nDCG.

(b) P@10.

Figure 4.28: Grain Comparison.

We have presented the application of the continuous evaluation framework using LongEval, a real evolving test collection. With each comparison analysis, we provide a method to compare systems evaluated using different epochs of an ETC. While the Pivot Comparison is dedicated to creating RoS, the projection strategy proposes the expected performance of a system in the next epoch, and the grains comparison proposes to compare the standardized performance of systems across different epochs.

4.5 Discussion

The Continuous Evaluation Framework is built in three stages: (i) a comparability validation step, (ii) a comparison strategy step, and (iii) a longitudinal analysis step.

The experiments with Robust$_e$ and TREC-COVID$_e$ show that the epochs of an ETC are comparable in a high proportion, enabling a meaningful comparison of systems evaluated in such epochs. This result also validates the creation of the time-based ETC, built with 90% of overlapped documents in successive epochs. Previous experiments present that with 90% of document overlap between different test collections, the probability of having the same RoS between the compared test collections is larger than 90% for MAP, Rprec, bpref and ndcg metrics [122]. It makes sense to have a lower percentage of comparable epochs (80%) with TREC-COVID$_e$, as this is a time-based ETC and it presents more variable performance values than a non-time-based ETCs (as experimented in [122]).

The second step proposes three strategies to validate the comparison of systems across the ETC epochs, we discuss each strategy:

- The results show that any Pivot system is able to create a more correct ranking of systems than a baseline at each evaluation epoch. This result follows the same findings of the Pivot Strategy in non-evolving test collections [57, 96]. In these works, the pivot was validated not only by splitting the topics but also by splitting the document sets, showing that using a pivot, it is possible to improve the correctness of RoS evaluated in different test collections, compared to the RoS created with absolute values (baseline approach).

- In terms of a projection comparison. The standardization examples represented the main problem of creating projection functions. When the standardization functions are not bijective, then the inverse function is not computable. In this case, the projection function is defined as the minimum and maximum values in the standardization range. This decision led to an expected performance in a large range of values making the longitudinal analysis step less precise.

- The final comparison strategy is a grain definition. We propose to create grain based on the performance of the reference systems at each epoch. At each grain, we find a big proportion of comparable epochs for Robust$_e$ and TREC-COVID$_e$.

This result is in line with the state of the art. Cattelan and Mizzaro [27] have shown that it is possible to compare systems evaluated over test collections that use no common queries by using an appropriate query selection strategy and metric normalization. They showed that query selection impacts the correlation of the rankings of systems. In our proposal, we have relied on similar ideas to exhibit specific behaviors of an evaluated system at each ETC, finding comparable grains across epochs using reference systems. In this sense, Berto et al. [15] obtained results that support the hypothesis that, by taking special care, the few queries selected on the basis of a given system population are also adequate to evaluate a different system population as well.

The third and last step of the proposed framework is a Longitudinal Analysis, which application depends on the comparison strategy. We discuss the advantages and limits of each analysis:

- We propose a Continuous Ranking using a Pivot system. The focus of this ranking is approximate to the real ranking of systems considering two evaluated epochs. The Pivot is able to define which system outperforms the other when they are compared in two epochs agreeing with the real results more than ranking them according to their absolute values (baseline). One limit of the pivot is the use of one system across all the compared epochs. A second limit of the pivot is that the score of the tested systems is not expressed on the same scale as the performance metric. When computing $\mathcal{R}_{se}\Delta_M$ computed using the pivot performance values of each system, the difference scores are in relation to the improvement of the test system versus the pivot at each epoch. Therefore, the pivot is useful to rank systems but not to interpret the performance of systems across different epochs.

- The expected performance is useful to understand the change of one system in time, computing a $\mathcal{R}_{se}\Delta_M$ of one system across different epochs, as a real effect of the environment on the system's performance. The expected performance is able to predict if the performance of the system will improve or decrease in the next epoch. However, there is still room to create more precise expected performance ranges when the standardization functions are not bijective. In some cases, the improvement is predicted, and the difference between the two systems is also correctly projected, but it fails to predict precisely the next performance value (larger ranges or translated). We noticed in our experiments that the expected performance may be sensitive to the number of queries. While Robust has 249 queries and short expected ranges, TREC-COVID only has 50 topics, resulting in large ranges of expected performance. Another limitation is the number of common queries required to compute the global performance of a system using the projection functions. To tackle this issue, we could explore standardization and projection of different query sets, for example, using comparable grains proposed in the grain definition step.

- Using grains, the evaluation shows the performance of systems in specific yet comparable categories. This analysis makes it possible to pinpoint the epochs in which a system has different performances according to the respective grain. While standardization proposes very stable performance values across epochs, which makes it difficult to understand the performance differences when comparing systems, we use standardized metrics in specific grains to compare systems on a common performance range.

Overall, the Continuous Result Delta Evaluation Framework proposes a set of strategies to compute meaningful $\mathcal{R}_{se}\Delta_M$ for systems evaluated in different epochs of an ETC.

4.6 Conclusion

In this Chapter, we have presented our Continuous Result Delta Evaluation Framework, which proposes a three steps framework to analyze the performance of systems evaluated in an ETC and to compute meaningful $\mathcal{R}_{se}\Delta_M$ of systems evaluated in different epochs of the ETC. We validate each step of the framework using two simulated ETCs, Robust$_e$ and TREC-COVID$_e$. Finally, we show its application in a real ETC, LongEval.

Our framework is able to rank systems evaluated in different epochs of an ETC. The Pivot comparison strategy proposes a correct continuous ranking of systems, even if such systems are not evaluated in the same epochs. The framework also proposes a performance comparison with a focus on one specific epoch. The expected performance analysis projects the performance of a system evaluated in a different test collection to the epoch of interest, meaning that all the compared systems are on the same performance scale. With the same goal, the grain strategy proposes to compare the system's performance on a standardized scale using meaningful granularities.

There is still room to improve the continuous evaluation framework, with further experiments focused on some choices that need further investigation. Although they are justified, the validity of the hypotheses, the definition of the granularities, and the projection estimation remain to be explored. Nevertheless, we show that, when using adequate parameters and choices, our framework is able to describe precise behaviors of IR systems on simulated and acquired evolving collections.

Chapter 5

Conclusion and Future Work

In this work, we tackle the problem of continuously evaluating the performance of Information Retrieval Systems when the systems and test collections evolve. This is translated to the task of comparing different systems evaluated in different test collections. In chapter 2, we draw an overview of the elements and the limits of the current evaluation in Information Retrieval. One critical requirement is the use of one common test collection to compare the systems' performances, making it impossible to apply the classical evaluation paradigm to a continuous evaluation.

To build a continuous evaluation of information retrieval systems, we summarize our contribution in two main points:

- First, we define evolving test collections, including a formalization, simulation, and acquisition framework.
- Second, we propose a continuous result delta evaluation framework allowing to compare systems evaluated in evolving test collection relying on $\mathcal{R}\Delta_M$.

To create a continuous evaluation, first, we need a suitable test collection to understand the changes in IR systems across evolving collections. Consequently, the first contribution undertakes the fact that there is no suitable test collection to evaluate the systems in a continuous way. To tackle this problem, we propose two methods to create Evolving Test Collections:

- By simulation: To use the available resources in the literature, we propose to adapt existing test collections to simulate Evolving Test Collections. Our simulation strategy relies on the creation of epochs with a set of features and constraints that controls the evolution of the test collections across the epochs. After evaluating several ETC built from different strategies, we conclude that a time-based ETC is the most suitable one to evaluate a continuous evaluation framework. The time-based ETC presented more variable results with any metrics than the other ETCs.

- By acquisition: We proposed an acquisition framework to create test collections from a Web Search Engine periodically. The acquisition resulted in LongEval, an ETC composed of three epochs with more than a million documents and almost one thousand queries per test collection.

Using these ETCs, we are able to evaluate systems and compute different performance values that vary according to the evolution of such test collections. Our next contribution was dedicated to proposing a methodology that provides a meaningful comparison of such changing performance values.

To compare the performance of systems evaluated in different epochs of an ETC, we propose to measure result deltas $\mathcal{R}\Delta_M$. We began by defining three types of $\mathcal{R}\Delta_M$. These types define respectively:

- $\mathcal{R}_s\Delta_M$ when we compare two systems in the same test collection;
- $\mathcal{R}_e\Delta_M$ when we compare one system in two test collections; and
- $\mathcal{R}_{se}\Delta_M$ when both the systems and the test collections vary.

The most difficult problems arise for the $\mathcal{R}_{se}\Delta_M$: in such case, we propose a continuous result delta evaluation framework based on three steps: (i) comparability validation, (ii) comparison strategy, and (iii) longitudinal analysis to handle the fact that systems and test collections evolve. The framework relies on an evaluation over ETC epochs.

The first step proposes a quantification of the similarity of two epochs by a comparability validation according to a set of reference systems. This step represents the first step in the characterization of test collection differences, or $\mathcal{K}\Delta$. This step filters which epochs are meaningful to compare across the epochs and represent a limit in the proposed framework when the epochs are not comparable. Therefore, more meaningful $\mathcal{K}\Delta$ measurements can inform in a better way the comparability between epochs.

The second step proposes different methods to define how the performance of the system can be compared through the transformation of the performance values using one Pivot system or a set of reference systems to compute standardization and projection function or determine comparison grains. This step defines how the $\mathcal{R}_{se}\Delta_M$ values can be computed in the third step of the framework. It is important to notice that, even if the three strategies led to similar results, the transformation of the performance values into a common and comparable scale, we decide to propose different solutions to cope with each alternative's specific problems. First, the pivot strategy relies on one system, which has to be replaced if its ability to create a correct RoS decreases, but this approach is very simple to apply and presents good results to create an RoS considering epochs with different queries and document sets. Second, the projection functions rely on a set of common queries across epochs, which is a strong limitation in an evolving setup.

As presented in LongEval ETC, the overlap of queries is smaller than the overlap of documents. Nevertheless, the projection functions rely on a set of systems that can change across epochs. Third and last, the quality of the grain comparison relies on the defined categories and the number of queries that exist in each of them. However, the grains are easy to define and adapt to the evaluation requirements of the framework user.

The third step proposes to perform a longitudinal analysis of the transformed performance values. Different analyses are implemented according to the strategy defined in the second step. Our experiments showed that all the analyses are able to present the evolution of the performance values considering the differences between the test collections at each epoch. Specifically, the continuous ranking compares the change in the position of the systems; the expected performance proposes a projected performance of a system into a specific epoch of interest, understanding how the performance of such a system may be impacted by the changes in the collection; and third, the grains present a standardized performance comparison at specific grains. Using the longitudinal Analysis, we can compute $\mathcal{R}_{se}\Delta_M$with the transformed performance of the systems.

As an additional contribution, presented in Appendix A, we created and shared a Web visualization tool to perform an exploratory analysis of IR systems evaluated in different epochs (or evaluation rounds). This tool is necessary to communicate and understand the results of a continuous evaluation as the analysis starts to become more complex and hard to interpret. The tool is designed as an open-source web tool that can be modified and adapted to specific requirements.

The described contributions provide the resources and methodologies to implement a continuous evaluation of IR systems and define how to compute $\mathcal{R}_{se}\Delta_M$ between systems evaluated across epochs of an evolving test collection.

Short-term Future works

Regarding the simulation of ETC, our proposal can be extended by considering several elements of the test collections jointly so that we may detect dependencies between elements, as in [44]. The definition of the mapping feature can integrate other aspects of documents and queries, for example, some change in the semantics or use of words, in order to compute other kinds of evolution in test collections.

LongEval provides an important resource to the IR community to analyze different aspects of IR systems and evaluation methodologies beyond continuous evaluation. For example, LongEval can be used to evaluate the temporal persistence of IR systems performance using reproducibility metrics as proposed in the CLEF LongEval-Retrieval task by the IRC team [66]. Other investigations can exploit the temporal characteristics of LongEval. From a different perspective, LongEval ETC needs to be evaluated in terms of its application and scalability in a Web search engine. LongEval was released in one and two months (as short-term and long-term test collections), and the resources from

part of Qwant were acquired monthly. The UGA team validated each release of the test collection in terms of the quality of the extracted queries and the number of relevant assessments. Including the acquisition and validation process in the development process of an industrial search engine needs to be studied in detail in the future to understand the limits of the proposed acquisition method.

From a short term-perspective, we identify that the continuous evaluation framework can continue to be improved at each comparison step. As a future general work, we propose to study the robustness of our framework in terms of the number of reference systems that are required at several steps of the framework. In terms of the Pivot comparison, further investigation can define the relationship between the evaluated systems and the Pivot used to compare them to decide if there exists a good Pivot and according to which features. In terms of the projected comparison, there is space to propose new ways to define the projection functions and the computation of expected performance, especially when the standardization functions are not bijective. In terms of the grains comparison, we acknowledge that further investigation can be generated in terms of defining automatic grains using more complex features than the performance of queries.

Finally, we propose to include new visualizations of the developed evaluation tool, with the goal of communicating and complementing the results of the continuous result delta evaluation. The availability of the tool can also be extended as a software-as-a-service tool to facilitate access to an on-running system.

Long-term Future works

Our long-term research perspectives focus towards building an explainable and interactive IR continuous evaluation framework and to integrate it into future adaptable IR systems.

The $\mathcal{R}_{se}\Delta_M$ measurements, using our Continuous Evaluation Framework, describe when the performance of a system is improving given different test collection epochs. We propose to include $\mathcal{K}\Delta$ to the explainability schema for the system's performance into the continuous evaluation framework. We are especially interested in explaining what is making the system fail with an analysis focused on the test collection characteristics. The first efforts are presented in two works, where the $\mathcal{K}\Delta$ tries to predict $\mathcal{R}_{se}\Delta_M$ in the context of ETCs. The objective is to compute the impact of test collection changes on the performance change of a system. As a first approximation, we rely on state-of-the-art Query Performance Prediction features as $\mathcal{K}\Delta$ [55]. Later, several $\mathcal{K}\Delta$s are quantified by means of TF-IDF and Language Models (LM) representations. An SVM classification model predicts $\mathcal{R}_{se}\Delta_M$ for various IR systems to explain the relation impact of each $\mathcal{K}\Delta$ into the $\mathcal{R}_{se}\Delta_M$ prediction [41]. These experiments have shown that $\mathcal{K}\Delta$ can predict the $\mathcal{R}_{se}\Delta_M$ values (with F1-score higher than 0.8 with MAP metric when using $\mathcal{K}\Delta$ as LM representations). This represents a first step of an explainable continuous evaluation, where we can describe which differences in the test collections impact the performance of systems. The second step is to bring the framework to scale by using it within an

industrial search engine.

More globally, we can explore in the future strong integration of explainable continuous evaluation frameworks into an IR system that can adapt its model and implement modifications according to the changes in the environment. This could pave the way to a new generation of IR systems, able to support automatic adaptation by measuring how the changes of the documents, queries and assessments (measured by $\mathcal{K}\Delta$) impact their performance (measured by $\mathcal{R}\Delta_M$).

Funding Acknowledgments

This work was supported by the ANR Kodicare bi-lateral project, grant ANR-19-CE23-0029 of the French Agence Nationale de la Recherche, and by the Austrian Science Fund FWF, grant I4471-N.

Appendix A

Continuous Evaluation Tool

We present here a visualization platform designed to implement exploratory data analysis in the context of a continuous evaluation of IR systems.

A.1 Introduction

We introduce Cont-Eval [56], an open-source software tool that targets researchers and search engine practitioners who repeatedly evaluate their information retrieval systems. Cont-Eval is designed to facilitate the continuous evaluation of information retrieval systems, enabling users to gain insights into the performance of their systems and identify areas for improvement. Our tool addresses this issue by proposing a set of visualizations that considers the performance changes across evaluations.

We recognize that evaluation is a repeated task. As the results consider multifactorial elements, such as the number of queries, documents, systems, and corpus, plain visualization becomes more challenging to program, visualize, and interpret. Our system addresses this challenge by providing an accessible and modifiable tool to visualize the evaluation of information retrieval systems continuously. By providing an interactive and dynamic dashboard that allows users to monitor their systems and track progress over rounds, we hope to help researchers and practitioners identify trends and insights that can inform the development and improvement of their information retrieval systems.

A continuous evaluation is based on repeated evaluation rounds that use different test collections. We propose using an exploratory visualization approach that considers how the performance of information retrieval (IR) systems changes across test collections, using two evaluation methods: standardization and meta-analysis.

Standardized scores can compare systems across different test collections without worrying about topic hardness or normalization [99]. Webber et al. [136] used a non-linear transformation that assumes standard normal distributions per topic on evaluation measures. Later, Sakai [99] proposed a linear transformation of the standardized scores as-

suming a uniform distribution. We present all of our graphics using either raw scores or standardized scores. We transform the raw performance values through a cumulative distribution function (CDF) that assumes a normal or uniform distribution, with the distribution parameters computed from the performance values of a set of baseline systems.

The work of Soboroff [116] proposes to use meta-analysis to evaluate a single system over multiple test collections, including different topics and corpus settings. Meta-Analysis helps to interpret the evaluation of systems tested over multiple collections, where one baseline is compared to a treated system resulting in a delta measure over multiple collections, generating a final mean difference with a confidence interval from the treated and baseline system. We follow his approach to compare the evaluation of a system across several evaluation rounds.

Understanding why the performance of an information retrieval system changes across different collections is essential to improve the system's quality [54]. Our visualization tool proposes to carefully analyse the performance of a system in the context of state-of-the-art research on evaluation, to explore in detail how and why the system's performance changed. The code and demonstration of our system are available at https://github.com/gabrielanicole/ExCEIR.

A.2 Related Work

Visualization of the results and of systems comparison have been studied in IR for a long time. In 1999, [95] proposed some ways to look at the dispersion of evaluation results on several query sets. In some way, this work considered several test collections at one time, but the process on these data was very limited.

Vis-Trec [120] proposed to display graphics from trec_eval generated results. It proposes query help-hurt and per-query difficulty displays. It also allows to compare one system against others but does not support per-query manipulations, and does not consider multiple rounds. [2] proposed to display the results of several systems using trec_eval results, by providing a display of evaluation measures at several top-k cutoffs. Unlike our tool, it does not integrate any filtering of systems and queries, there is no explicit comparison view for one specific system and does not consider multiple rounds.

RecDelta [31] presents an interactive tool for cross-model evaluation of top-k recommendations. The user can visually compare the performance of various recommendation algorithms and their recommended items. The main functionality is the visualization of distribution δ scores between results. This tool helps to explain the results of a system using two visualizations: Venn diagrams and HeatMap. In the same line, DiffIR [65] presents a visualization tool to show the difference between retrieved elements from different ranking systems. The focus of DiffIR and RecDelta is the specific retrieved items of each system, while for us, the comparison is performed in terms of differences in the

performance metrics across evaluation rounds.

In contrast to these existing visualization tools, our tool provides a visualization for exploratory analysis of several evaluation rounds. It implements standardization score visualization and meta-analysis to deal with changes in the systems' performance. It uses trec_eval evaluation outputs. This makes it easy to integrate with evaluation results. Finally, the tool provides a comprehensive view of the changes in the system's performance over time.

A.3 Architecture

Our open-source software facilitates continuous evaluation of information retrieval systems. The software is built on free and open-source technologies and uses Django[1] version 4.1.6 as the main web framework.

The front-end includes Jquery and D3.js[2], a JavaScript library for creating dynamic and interactive data visualizations in web browsers. This enables the system to be easily accessible and modifiable. D3.js creates scalable vector graphics (SVG) in the visualization tool. The extensive library makes it easy to develop new graphics that can be integrated into the system. In the back-end, Python3.10 is used as the programming language. The system is based on four data science libraries: Pandas [70] and NumPy [125] for data manipulation, and SciPy [1] and StatsModels [112] for computation of evaluation metrics, including standardization and meta-analysis.

To organize the data, and following the Django framework, we store the evaluation scores of the system considered, computed by trec-eval, on the static directory of our application. The results of each round are grouped in the same subfolder.

A.4 Functionalities

Continuous evaluation of information retrieval systems is a complex task that requires comparing the performance of multiple systems across multiple rounds. To facilitate this process, our tool allows users to select a set of baselines to compare a tested system's performance across rounds. This enables users to track the improvement of their system over time and identify areas for further development and improvement. The philosophy behind selecting a test system and a set of baselines is based on real IR systems. In a search engine that is constantly changing, the final version of the search engine is compared to older versions of the same system, these older versions become baselines, and it is expected that the applied modifications to the system will improve its performance.

[1] https://www.djangoproject.com/
[2] https://jquery.com/ & https://d3js.org

To compare the performance of the test system and the baseline, our tool features three main views of the data: Rounds, Queries, and Systems. In the three views, users can choose to run the visualization on the full set or a subset of queries, systems, or rounds.

The main view is the **Round** view, where an exhaustive analysis of the systems' performance is developed, starting from an *Overview* of the systems' performance across rounds in different standardized scales; followed by the comparative analysis of the test systems vs the baselines called *Delta Evaluation*; and finally, a *Meta-Analysis* to conclude if, according to the performance of the systems in multiple rounds, the test system (i.e., the last version) is better to the baselines (i.e., the older versions of the search engine). In this case, the performance of the system is represented as the mean value of the selected metric. As the goal of a meta-analysis is to compare the performance of a system in different rounds, this graphic is not supported in the Queries and Systems views.

Queries and **Systems** views are specific cases of the **Round** view. In the **Query** view, *Overview* presents the results of the evaluated systems in one selected round or the results of one system in different rounds. Therefore, it is possible to select only a single round or only a single system. Using the same selection, *Delta Evaluation* shows the performance difference of the selected systems per query (if several systems are selected) and the performance difference of each query versus the final round (if several rounds are selected).

In the **Systems**, *Overview* presents the performance of the test system and the baselines in different rounds, using the mean value across queries. In *Delta Evaluation*, we present the performance difference of each system in the selected rounds versus the last one. With this visualization, it is possible to identify pair of rounds that provides the most different results for a specific system.

As detailed above, the three views work with the dimensions of the evaluation: round, system and query; for clarity, each view presents only two of these dimensions, while the third is aggregated. In summary:

- In the **Rounds** view, users can compare for each round the metric mean over the selected set of queries, for a selected set of systems. The x-axis in the graphics of this view always represents the round number.

- In the **Queries** view, users can compare the performance of a set of systems for each individual query within a specific round. The x-axis of this view represents the query id.

- The **Systems** view presents the performance of different systems across various rounds for a selected set of queries. This view is used to visualize the performance change of the system across the rounds. The x-axis of this view represents each system.

A.5 TREC-COVID evaluation Example

We present an example of the functionalities of our visualization tool using the TREC-COVID test collection [127]. TREC-COVID is a test collection organized in five rounds, where each round is composed of a specific release of CORD-19 document collection [134], a set of incremental topics, and relevant judgments. In the example below, we compare the performance of six classical IR systems across the five rounds.

A.5.1 General View

Figure A.1 presents the organization of the three views. In general, a view is composed of a summary section, a set of filters and a set of visualizations that considers the selected parameters of the filters.

The summary section presents the total number of rounds, systems and queries in the application (extracted from the static/evaluation folder). In the TREC-COVID case, we have 5 rounds of evaluation, we did evaluate six systems in each round and there is a total of 50 queries evaluated.

In the filter section, users can filter and select the evaluation rounds, systems, and queries that they want to analyze. Within the tool, the selected system is referred to as the *test system*, while all other systems are considered *baselines*.

In the example, we select *TF_IDF* as the test system, leaving 5 systems as baselines. To run the evaluation, it is required to select one metric from the list of metrics available in the files computed with trec_eval: in the example case we select "*map*". If the value of any of the filters changes, it is necessary to press the "run evaluation" to update the visualizations.

In this example (Figure A.1), we present the **Rounds** view. Here, the visualizations section presents:

- an *Overview* to display the performance of the test system along with the baselines;
- a *Delta Evaluation* that *compares* the improvement of the test system versus the baselines;
- and finally, the results of the *Meta-Analysis* that compares the results of a selected baseline versus the test system. In this example, we will see how the graphics are related and have a specific purpose that leads to a decision that is implemented for the next graphic.

A.5.2 Overview

Figure A.2 shows an overview of the systems' performance across rounds. Each bar represents the mean value of the selected queries on the selected rounds, we have selected all

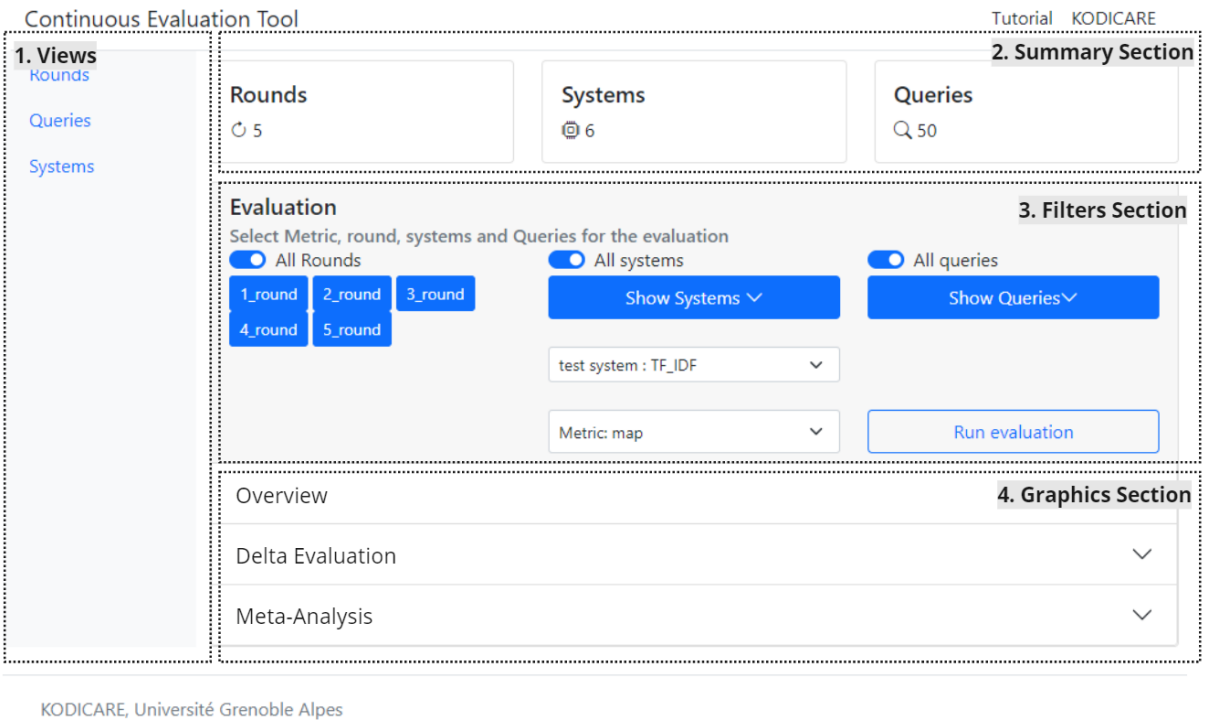


Figure A.1: Initial Page: Users start by selecting which systems they want to evaluate, and the evaluation settings: rounds, queries, metric, etc.

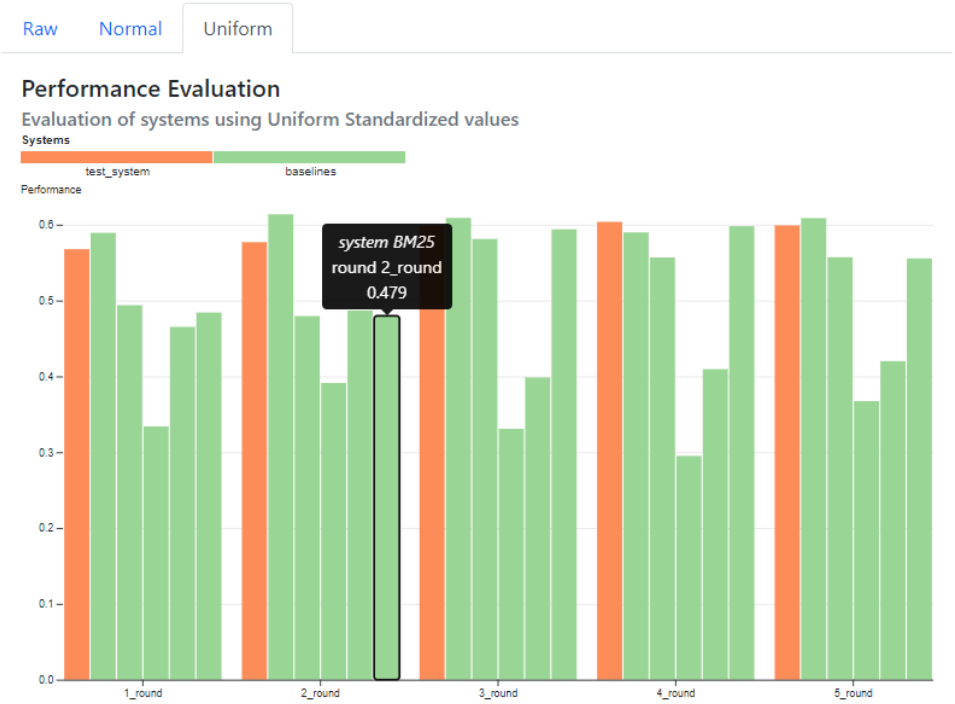


Figure A.2: Overview in the *Rounds* view. It shows the performance of the test system versus 5 baselines at each round of TREC-COVID.

the queries and rounds for this example. The performance of the test system is presented in orange and the baselines are in green. The performance values could be presented as *Raw values*, or we can compute *standardized scores* according to the baselines. We implement score standardization assuming *Normal* or *Uniform* distributions.

Figure A.3 presents the same graphic in the Queries view. In this case, it is possible to see how the performance of the test system (TF_IDF) and one baseline system ($BM25$) changes query by query in the first round of TREC-COVID. In this example, we present the scores using Uniform standardization.

A.5.3 Delta Evaluation

From the performance evaluation of the *Overview Performance Evaluation* shown in Figure A.2, users can go deeper and observe the relative performances of the systems. The *Delta Evaluation* view allows us to visualize compared performances of systems for a specific score standardization method. It shows the relative performances of the test system with baselines for each round.

Delta Evaluation graphic focuses on comparing the test_system versus each baseline. In this case, we compute the performance difference between systems. Figure A.4 presents

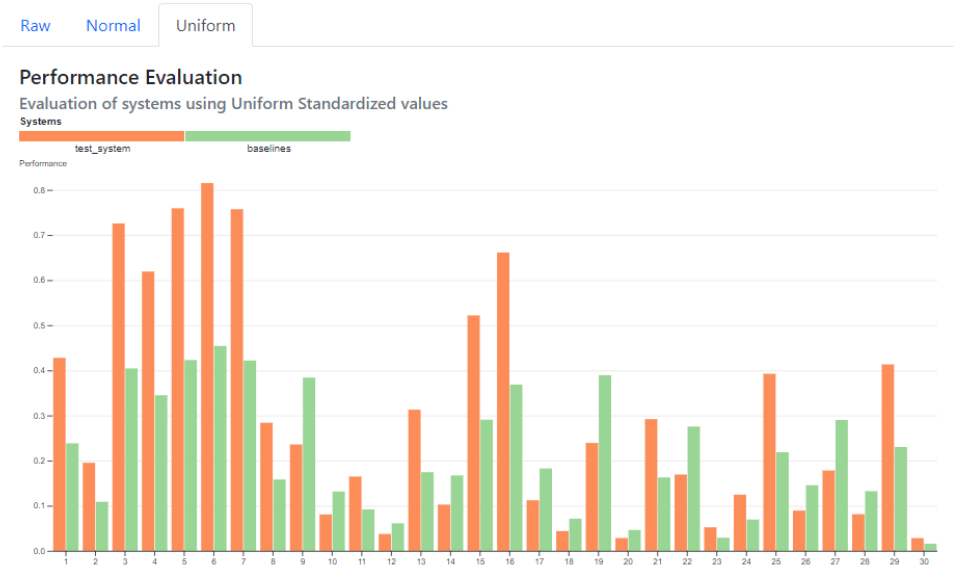


Figure A.3: Overview in *Queries* view for a selection of queries for Round 1 of TREC-COVID. It shows the performance of the test system vs one baseline for each query.

the comparison of *TF_IDF* in each round of data. Each bar represents a specific baseline, which is red if the mean performance of the test system is worse than the mean performance of the baseline and blue otherwise.

A.5.4 Meta-Analysis

The final visualization section is dedicated to presenting the results of a Meta-Analysis across the rounds. The goal is to extract an overall conclusion of the improvement of a test_system versus a specific baseline across the evaluated rounds. In this example, and considering the results of the Delta Evaluation graphic, we select *BM25* as the baseline.

Figure A.5 shows a forest plot, traditionally used to present the results of Meta-Analysis [116]. Each line represents the effect of test_system versus the baseline at a specific round. The last line shows the random effects, which is the final estimation of the effect size of the test system versus the baseline in all the rounds. In our TREC-COVID example, the test system *TF_IDF* has a positive effect versus *BM25*.

A.6 Conclusion

In this work, we present Cont_Eval, a tool that facilitates continuous evaluation of information retrieval systems by visualising performance metrics across rounds. We propose a set of visualizations from three perspectives: rounds, systems and queries. We also

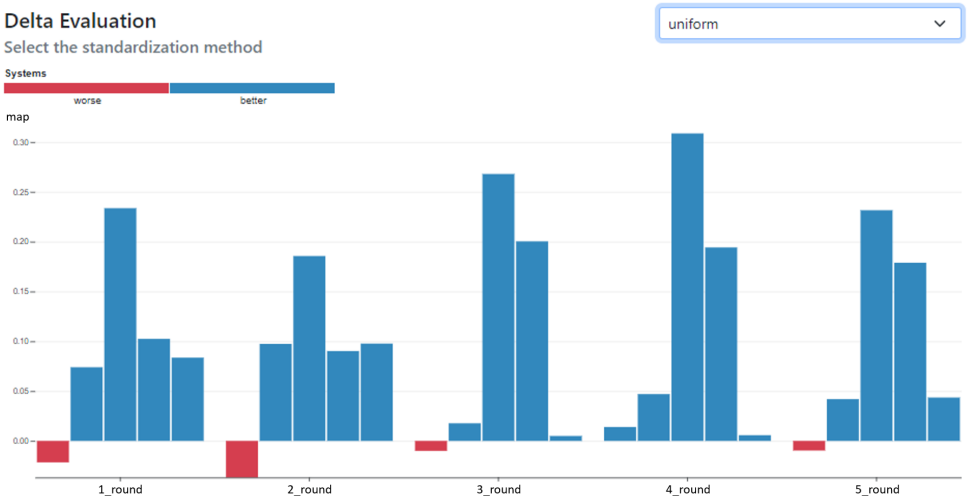


Figure A.4: Delta Evaluation for test system vs baselines for each round.

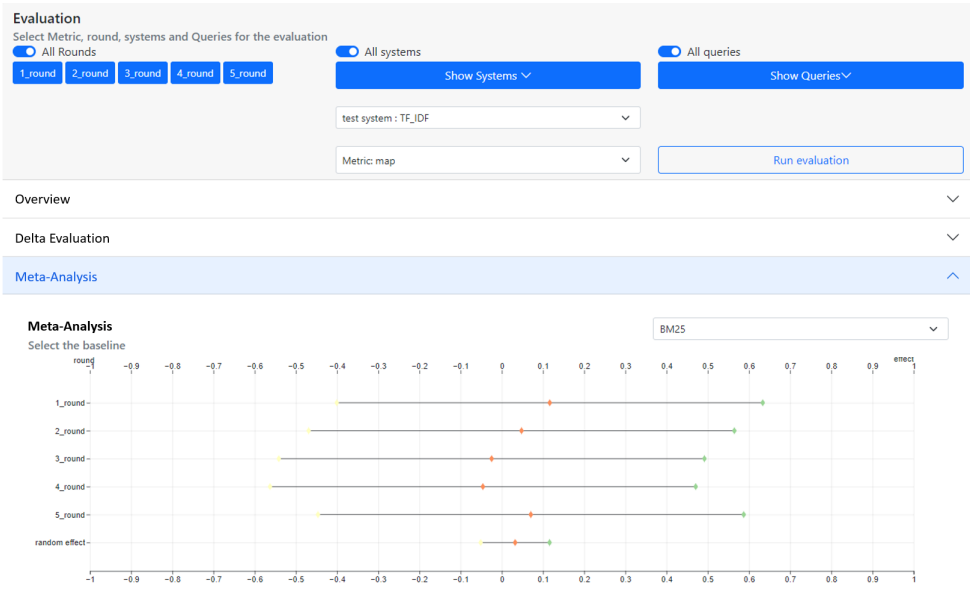


Figure A.5: Meta-Analysis of *TF_IDF* System versus a *BM25* baseline.

include state-of-the-art evaluation methods such as standardization and meta-analysis to help the interpretation of the performance changes across evaluation rounds.

In the future, we plan to enhance the visualization tool by incorporating metrics that take into account the changes of the test collections across evaluation rounds, as in [57]. Additionally, we will include metrics to guide the selection of standardization methods and baselines, and we will include the empirical standardization method proposed by Urbano et al. [123]. Furthermore, we aim to expand the capabilities of the Queries and System views by introducing new visualizations that describe changes in the queries and system results.

Appendix B

Résumé en Français

L'évaluation classique des systèmes de recherche d'information repose sur une *collection de tests statique*, composée d'un ensemble de documents, d'un ensemble de requêtes et de jugements de pertinence. L'évaluation permet de mesurer les performances d'un ou plusieurs systèmes. Toute modification de l'un des éléments de l'évaluation, comme l'ensemble de documents, les requêtes, les jugements de pertinence ou les systèmes évalués, a un impact sur la mesure des performances [45, 133].

Dans le cas de la recherche sur le Web, l'environnement (documents, requêtes) est en constante évolution. Utiliser une collection de tests statique n'est pas adapté à cette réalité changeante, et une collection statique serait insuffisante pour évaluer sur le long terme un système industriel de recherche d'information. Par conséquent, l'évaluation hors ligne des systèmes de recherche d'information ne peut pas être directement appliquée aux systèmes de recherche en direct utilisés dans l'industrie : par exemple, mener une campagne une fois par an n'est pas suffisant ; ajuster constamment un système à un référentiel peut conduire à sur-adapter le référentiel plutôt qu'à améliorer les performances du système, qui doivent être valides pour différentes collections de tests.

On a donc besoin de mettre en place un cadre pour évaluer de manière adéquate et continue les systèmes de recherche. Par rapport à l'évaluation classique de la RI sur des collections de tests statiques, l'évaluation continue des systèmes de RI intègre la notion de "temps" dans le processus d'évaluation. En plus d'autres aspects, une évaluation longitudinale d'un système de RI vise à vérifier si la qualité d'un (ou plusieurs) système(s) est meilleure, aussi bonne, ou pire, au fil du temps. Plus précisément, l'évaluation longitudinale est capable d'identifier certaines raisons du changement de comportement d'un système. Dans ce travail de recherche, nous cherchons à répondre à la question suivante :

- *Comment comparer les performances de différents systèmes de recherche d'information lorsque les collections de tests évoluent ?*

La littérature ne fournit pas de collection de tests appropriée pour effectuer une évaluation continue des systèmes. Nous définissons le problème suivant :

- Il n'existe aucune collection de tests avec des évolutions contrôlées de l'ensemble de documents, de requêtes, ou de jugements de pertinence disponibles dans la littérature, dans laquelle nous pouvons appliquer et tester une évaluation continue des systèmes de recherche d'information.

Il existe cependant des collections de tests statiques qui pourraient être adaptées pour simuler l'environnement Web dynamique. La maintenance des collections de tests [115] propose une méthode pour mettre à jour une collection existante en incorporant de la dynamique à la collection de tests. Nous nous appuyons sur ces travaux pour créer des collections de tests évolutives et contrôlées qui peuvent être utilisées pour évaluer et comparer en continu les performances des systèmes.

L'évaluation des performances repose sur différentes métriques, et différentes méthodes ont été proposées pour déterminer si un système S1 surpasse un système S2. Le cadre de l'évaluation classique est très limitant pour mener une évaluation continue car il nécessite d'évaluer tous les systèmes comparés en utilisant la même collection de tests. Cela est problématique étant donné que les performances des systèmes sont affectées par les changements dans la collection de tests. Cette recherche aborde ce problème :

- Nous ne pouvons pas comparer les performances d'un système évalué dans une collection de tests à celles d'un système évalué dans une deuxième collection de tests, car les variations de performances dépendent des changements dans ces collections. Il n'existe pas de cadre d'évaluation permettant de comparer les systèmes évalués dans des collections de tests évolutives et qui explique également comment les changements dans l'environnement d'évaluation affectent les résultats de performances.

À notre connaissance, il n'existe pas de cadre d'évaluation qui interprète de manière continue les performances d'un système évalué dans différentes collections. La normalisation [99, 123, 136] aborde le problème de la comparaison de requêtes de difficultés variables, mais elle ne montre pas comment interpréter l'effet des différentes collections de tests dans l'évaluation de différents systèmes. Les méta-analyses [116] et les mesures de reproductibilité [18] se concentrent sur l'extraction d'un effet global des collections de tests, mais elles nécessitent d'évaluer tous les systèmes comparés pour chaque collection de tests. Cette configuration est différente de notre évaluation continue, qui aborde le problème d'une évaluation continue du Web où les systèmes et les environnements de test changent.

Pour répondre à notre question de recherche, nous proposons une Collection de Tests Évolutive pour évaluer les systèmes de manière continue, ainsi qu'un cadre d'évaluation continue qui définit comment comparer les performances des systèmes évalués dans de telles collections de tests évolutives.

Dans ce qui suit, nous présentons la Collection de Tests Évolutive dans la section B.1, l'évaluation continue dans la section B.2, et notre conclusion dans la section B.3.

B.1 Collection de Tests Évolutive

Nous définissons une collection de tests dynamique (DTC) comme une liste de plusieurs collections de tests qui présentent des variations entre elles dans le but d'évaluer un système dans un environnement changeant. Par conséquent, une DTC étend la collection de tests statique en définissant une séquence d'*époques* représentant chaque collection de tests, en faisant varier les composants dans les époques précédentes et suivantes dans la DTC :

Une **Collection de Tests Dynamique** (DTC) est une liste de collections de tests (TC), où chaque collection de tests TC_i comporte trois composants (ensembles) : les documents (D_i), les requêtes (Q_i) et les jugements de pertinence ($Qrel_i$).

Nous nous intéressons à définir une collection de tests dynamique spécifique : une Collection de Tests Évolutive, qui vise à soutenir l'évaluation de la RI lorsque l'évaluation, et donc la collection de tests, évolue : à chaque époque, les documents, les requêtes et les jugements de pertinence changent de manière *évolutive et cohérente*. Dans une collection de tests évolutive, nous nous attendons à ce que des caractéristiques pouvant être calculées dans les composants mesurent l'évolution de la collection de tests, par exemple, un ensemble d'éléments communs d'une époque à la suivante. Un autre facteur de cohérence important est l'existence d'un ensemble unique de valeurs d'évaluation (AV) à travers les époques, de manière formelle :

Une **Collection de Tests Évolutive** (ETC) est une DTC avec des documents, des requêtes et des jugements de pertinence évolutifs utilisant un ensemble commun de valeurs d'évaluation AV.

B.1.1 Simulation

Nous simulons une ETC de manière contrôlée. Pour pouvoir construire une (ou plusieurs) ETC, nous utilisons une simulation basée sur une collection source de tests statique TC^s et un ensemble de paramètres qui contrôlent l'évolution des composants de l'ETC sur n époques. Cette évolution contrôlée nous permet d'étudier précisément le comportement des systèmes évalués.

Nous définissons la simulation $\mathcal{S}$ pour créer une ETC simulée, avec les paramètres suivants : un TC^s, un nombre d'époques (n), la taille de chaque C_i (N_C), et une *stratgie* de simulation. Par conséquent, $\mathcal{S}$ est définie comme suit :

$$\mathcal{S} : (TC^s, n, N_C, stratgie) \rightarrow ETC$$

ensuite, une ETC simulée est décrite comme suit :

$$ETC_{\text{simulée}} = \mathcal{S}(TC^s, n, N_C, stratgie), \text{avec } AV = AV^s$$

La stratégie est définie comme suit :

- Fixer la **cardinalité** de certains composants pour qu'ils soient égaux.
- Fixer une valeur d'**overlap** globale o entre les composants successifs.
- Par exemple, une fonction d'**ordonnancement** FC peut exister sur un composant C_s.

Nous proposons deux instantiations de cette stratégie générale :

Random ETC : Une Random ETC imite un environnement changeant, par exemple, un ensemble de documents qui peuvent être supprimés et réintégrés ultérieurement dans la collection de tests, ou des requêtes liées à une saison spécifique. Cette évolution est simulée en extrayant aléatoirement plusieurs échantillons de la collection de tests TC^s. Pour créer un Random ETC, nous mettons en œuvre la stratégie de simulation comme suit :

$$stratégie = (cardinalité)$$

cette stratégie est uniquement basée sur la cardinalité, qui contraint le nombre d'éléments dans chaque C défini dans N_C.

Overlapping ETC : Lors de l'évaluation des systèmes à l'aide d'un Random ETC, comme nous ne contrôlons pas le degré de chevauchement entre les TC consécutifs, il est difficile d'évaluer le comportement d'un système sans savoir à quel point les collections de tests sont similaires. Par conséquent, le Random ETC simule des contextes changeants, mais pas une évolution. Une Ov. ETC simule l'évolution de la collection de tests en contrôlant la similarité entre les époques des composants, représentée par le chevauchement dans l'ETC : toutes les paires de TC consécutives de l'ETC sont également similaires par rapport à une caractéristique de similarité qui doit être définie. Pour créer un Overlapped (Ov.) ETC, la stratégie de simulation est la suivante :

$$stratégie = (cardinalité, overlap, \mathcal{F}_C)$$

La *stratégie* repose sur un ordonnancement complet des éléments des composants $\mathcal{F}_C$, en attribuant une valeur spécifique à chaque élément. Elle n'autorise pas la réinsertion d'éléments des versions précédentes et définit un nombre fixe d'éléments chevauchants entre les époques des composants (*overlap*). Tout comme dans le Random ETC, la *cardinalité* est également contrainte, N_C étant une valeur constante.

À l'aide des ETC simulées, nous pouvons créer plusieurs types d'ETC, par exemple en utilisant les sous-collections de pointe : le partitionnement et le bootstrap avec des ETC aléatoires. Il est également possible de créer un CTE à chevauchement basé sur le partitionnement (Ov. ETC), qui permet de contrôler la quantité de changement entre les époques successives définies par une valeur de chevauchement. Plus spécifiquement, nous nous intéressons à un ETC à chevauchement basé sur le temps, qui permet de contrôler les changements entre les époques avec une base temporelle afin de simuler un environnement Web dynamique.

B.1.2 Acquisition

Dans cette section, nous présentons LongEval [50], une nouvelle collection de tests en évolution avec des données acquises à partir du moteur de recherche Qwant. L'ETC LongEval est proposée pour soutenir l'évaluation des systèmes de RI Web commerciaux et open-source de pointe. Ce ETC est dédié à fournir une évaluation à grande échelle et est capable de faire face à l'évolution temporelle des données Web réelles. Plusieurs organisations ont contribué à la construction de LongEval. La principale contribution de cette thèse est la conception du processus d'acquisition et la définition des sujets qui guident la collecte de documents, de requêtes et de jugements de pertinence au fil du temps.

L'acquisition est périodique et se répète au fil du temps pour construire chaque collection de tests qui compose le ETC. Les requêtes, les documents et les jugements de pertinence changent d'une collection de tests à l'autre, mais ils sont acquis en fonction de leur relation avec un ensemble défini de sujets (qui sont des mots-clés décrivant un concept large et général). Cela est illustré dans la Figure B.1 :

1. L'acquisition d'un ensemble de **thèmes**, sélectionnés à partir du Web et des médias sociaux. Cette acquisition est basée sur des sujets tendance, mais stables à long terme, et n'est réalisée qu'une seule fois pour l'ensemble de la collection LongEval.

2. La sélection de **requêtes** de recherche, liées aux thèmes, provenant des requêtes réelles des utilisateurs de Qwant.

3. La création de **jugements de pertinence**. Nous nous appuyons ici sur deux approches : une approche implicite utilisant des modèles de clics [33] calculés à partir des logs de Qwant, et une approche explicite utilisant des jugements manuels, qui ont été réalisés après la soumission de ce manuscrit. Étant donné que chaque collection de tests contient plusieurs milliers de requêtes, des évaluations explicites seront effectuées sur un sous-ensemble limité de requêtes sélectionnées manuellement.

4. L'acquisition du **corpus de documents**. Ce corpus est composé de : i) tous les documents Web qui ont été affichés pour chaque requête d'une collection de tests,

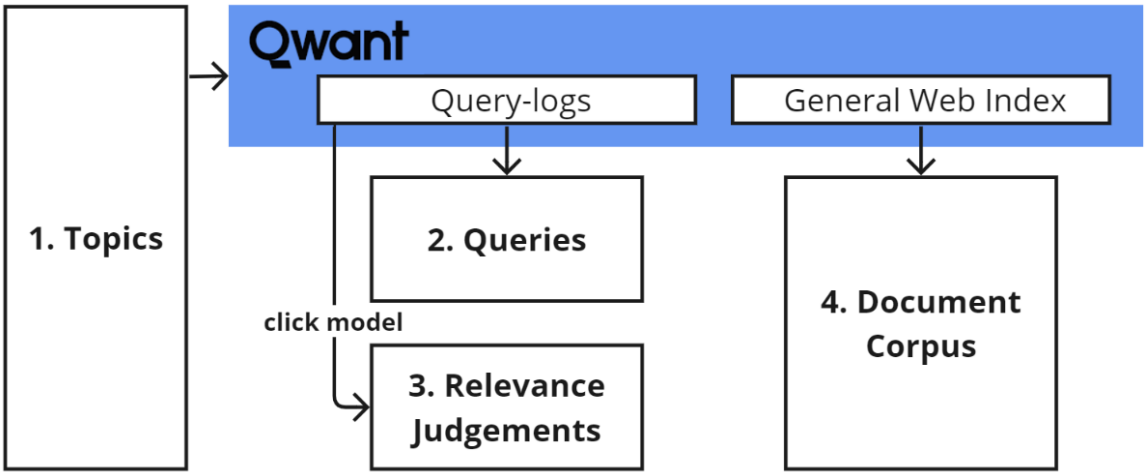


Figure B.1: Stratégie générale d'acquisition de données. Les ressources privées de Qwant sont en bleu. Les boîtes blanches représentent les éléments de la collection de tests.

et ii) un échantillon aléatoire important de l'index de Qwant. Ce protocole conduit à un corpus contenant un mélange de documents pertinents et non pertinents. Le processus présenté gère l'évolution des pages Web, car le corpus est composé non seulement des URL, mais aussi du contenu des pages Web acquis à un moment spécifique.

Comme décrit précédemment, LongEval est conçu pour évaluer les systèmes dans le temps. Pour ce faire, l'acquisition présentée est réalisée périodiquement, généralement chaque mois. À chaque période de temps t, nous créons une collection composée des requêtes, des jugements de pertinence et des documents collectés au cours de ce mois. L'ensemble complet de LongEval, composé d'une séquence de collections, évolue donc de manière dynamique. Cela nous permet de créer et de fournir des collections de tests pour différentes périodes de temps.

L'acquisition temporelle est présentée dans la figure B.2. L'acquisition est réalisée en trois périodes : un temps t, un second temps t' en tant qu'acquisition à court terme, et une troisième période d'acquisition t'' en tant qu'acquisition à long terme.

Enfin, nous présentons la structure finale de LongEval ETC :

$$\text{LongEval} = (TC_{june}, TC_{july}, TC_{sept}),\ AV = \{0, 1, 2\}$$

Nous répétons ce processus de collecte de données sur plusieurs mois et créons la collection d'entraînement t ($t = Juin$) ainsi que deux collections de tests t' ($t' = Juillet$) et t'' ($t'' = Septembre$). La figure B.3 montre LongEval ETC, comprenant les trois collections de tests :

La collection de juin est utilisée comme collection d'entraînement dans CLEF LongEval.

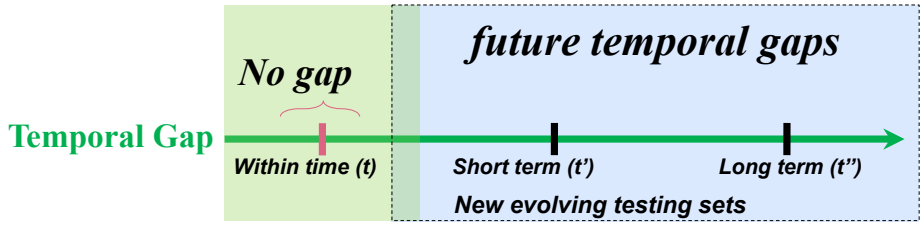


Figure B.2: Cadre acquisition global pour la collection de test LongEval [4].

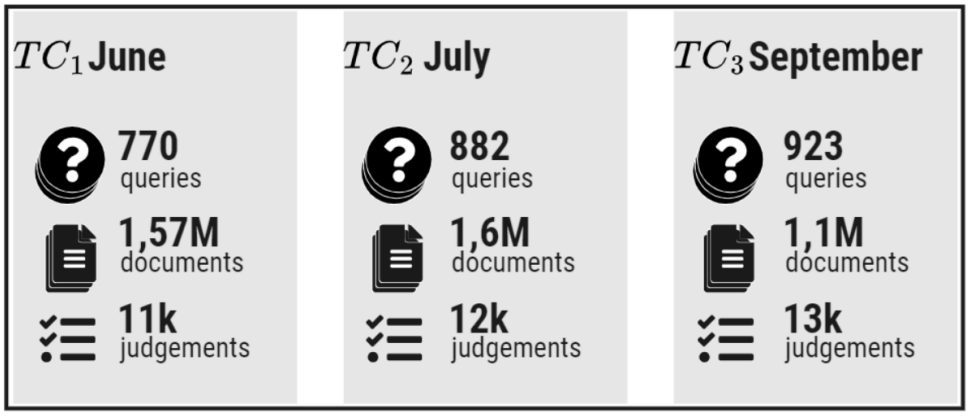


Figure B.3: LongEval ETC.

Elle a été collectée en juin 2022 et mise à disposition sur l'infrastructure Lindat[1]. Les collections de juillet et septembre, collectées en 2022, sont disponibles sur l'infrastructure Lindat[2] en tant que collection de tests LongEval.

B.2 Évaluation continue

Nous définissons l'évaluation continue comme suit :

Definition 13 *L'évaluation continue est une tâche consistant à comparer les performances d'un ou plusieurs systèmes dans une collection de tests en évolution.*

Dans un cadre d'évaluation continue, il est essentiel de prendre en compte les changements qui se produisent au fil des époques de la collection de tests en évolution ainsi que les modifications apportées au système de recherche d'information lui-même, car ces facteurs peuvent avoir un impact significatif sur l'évaluation d'un système mesurée par une métrique de performance M. Il est donc essentiel de pouvoir mesurer avec précision

[1] http://hdl.handle.net/11234/1-5010
[2] http://hdl.handle.net/11234/1-5139

l'impact des modifications apportées à la collection de tests, ainsi qu'aux systèmes, sur les résultats de l'évaluation.

Pour comparer les performances des systèmes évalués de manière continue à l'aide d'une métrique de performance M, nous proposons les **différences de résultats** ($\mathcal{R}\Delta_M$), définies comme suit :

Definition 14 *Une différence de résultats $\mathcal{R}\Delta_M$ est la différence mesurable de performance, selon une métrique de performance M, entre deux systèmes évalués sur deux époques successives d'une collection de tests en évolution, où les systèmes et les composants de la collection de tests peuvent évoluer.*

Étant donné qu'une collection de tests en évolution est composée de plusieurs époques ($TC_i \in ETC$), nous proposons une comparaison par paire des performances des systèmes.

Comme les systèmes sont évalués à des époques différentes, la comparaison tient compte de la performance des systèmes évalués sur des collections de tests différentes, et se fier uniquement aux valeurs absolues des métriques, comme décrit dans la section 2.2.2.1, n'est pas réalisable. Par conséquent, nous proposons de mettre en place un cadre d'évaluation pour calculer les différences d'évaluation sous forme de $\mathcal{R}\Delta_M$, en comparant $S1$ évalué dans TC_1.

Notre objectif est de calculer $\mathcal{R}_{se}\Delta_M$, mais il est difficile de le mesurer directement, car les deux systèmes ne sont pas directement comparables : à la fois les époques de la collection de tests en évolution et les systèmes sont différents. Pour obtenir une estimation de cette mesure, nous proposons :

Definition 15 *Un cadre d'évaluation continue des différences de résultats ($\mathcal{R}_{se}\Delta_M$) qui définit comment calculer $\mathcal{R}_{se}\Delta_M$ à partir d'une analyse longitudinale des valeurs de performance transformées des systèmes évalués dans différentes époques d'une collection de tests en évolution.*

Le cadre d'évaluation continue doit définir quand il est possible de comparer des systèmes évalués à différentes époques, puis comment transformer les valeurs de performance de tous les systèmes évalués à différentes époques afin de les comparer dans une analyse longitudinale.

Grâce à la définition de l'évaluation continue des différences de résultats, nous sommes en mesure de proposer un cadre d'évaluation utilisant des collections de tests évolutives. Notre cadre d'évaluation repose sur trois étapes réalisées en séquence (voir Figure B.4) :

Étape 1. La Validation de Comparabilité (CV) est une étape décisionnelle qui détermine si les époques successives sont *comparables*, c'est-à-dire si les différences entre elles ne sont pas trop grandes.

Étape 2. Une étape de Stratégie de Comparaison (CS), dans laquelle nous définissons la stratégie pour transformer la performance du système en une *échelle commune*.

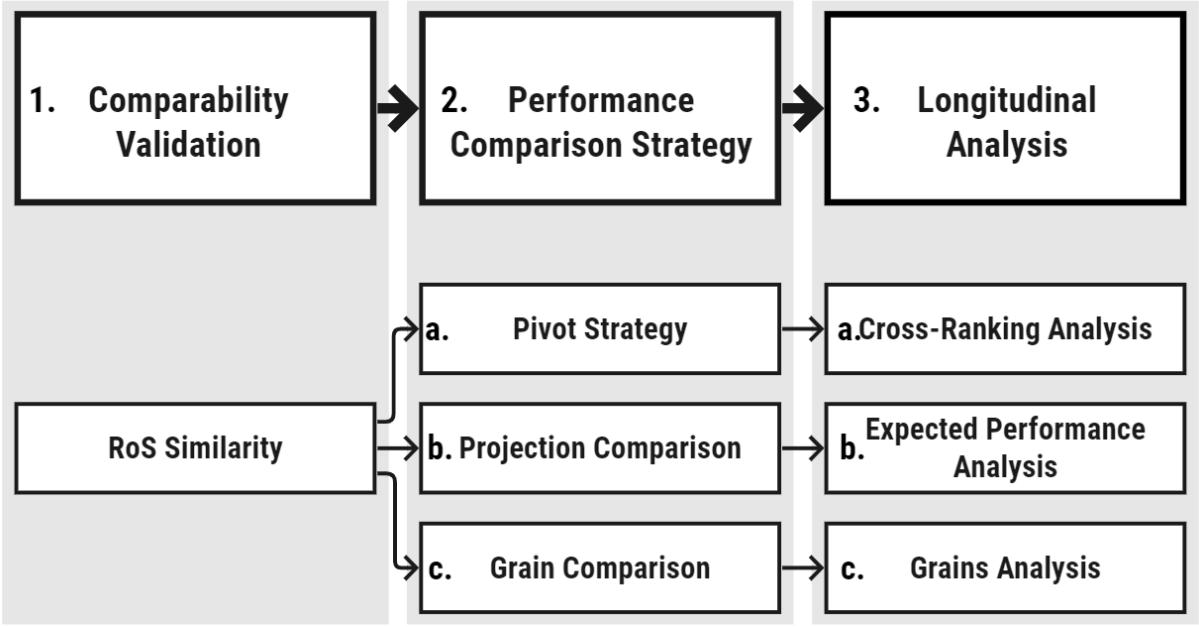


Figure B.4: Cadre de l'Évaluation Continue des Deltas de Résultats.

Elle peut être mise en œuvre par (2.a) une Comparaison *Pivot*, par (2.b) une Comparaison *Projection*; et par (2.c) des Comparaisons par *Grains* :

2.a Une **Stratégie Pivot** (PIV) définit un système en tant que pivot pour créer un classement des systèmes en incorporant l'évaluation du pivot sur les époques comparées ;

2.b Une étape de **Projection** (PRO) définit des fonctions de projection pour transformer la performance d'un système d'une époque à une autre. Les fonctions sont calculées à l'aide d'un ensemble de références et d'un ensemble de requêtes communes aux époques ; ou,

2.c Une étape de **Définition de Grains** (GRA) définit des groupes de requêtes sur les époques en fonction d'une caractéristique de regroupement, afin de calculer une performance normalisée dans ces groupes.

Étape 3. Une étape d'Analyse Longitudinale (LA) qui permet l'évaluation continue des systèmes sur plusieurs époques successives. Selon la méthode utilisée, une analyse spécifique est réalisée, à partir de laquelle $\mathcal{R}_{se}\Delta_M$ peut être calculé :

a. L'analyse du **Classement Continu** compare le pivot aux systèmes testés pour créer un classement des systèmes à chaque paire d'époques.

b. L'analyse des **Performances Attendues** calcule une performance attendue en utilisant la projection d'un système dans une époque précédente, et la compare à la dernière époque évaluée.

c. L'analyse des **Grains** compare la performance des systèmes par *grains significatifs* de différentes requêtes de deux époques.

Par conséquent, l'Évaluation Continue des Deltas de Résultats se compose de trois étapes, ce qui donne une liste de valeurs de performance pour chaque système évalué sur l'ensemble des époques de l'ETC. Ainsi, $n-1$ valeurs $\mathcal{R}_{se}\Delta_M$ peuvent être calculées, une liste pour chaque paire de systèmes testés.

$$\text{Cont}\mathcal{R}\Delta_M\text{Eval} : (CV, CS, LA, ETC) \rightarrow \mathbb{R}^n \tag{B.1}$$

où chaque étape CV, CS et LA a un ensemble de paramètres respectifs selon la méthode spécifique décrite, qui sont détaillés dans les sections suivantes.

Notre cadre de travail est censé fournir des indications précises sur le comportement d'un ou plusieurs systèmes sur une collection de tests évolutive dans des conditions strictes qui conduisent à des résultats significatifs : nous évaluons la comparabilité des époques, nous évaluons la correction de la stratégie pivot, et nous évaluons la validité de la granularité utilisée pour évaluer les systèmes.

Notre cadre de travail est capable de classer les systèmes évalués dans différentes époques d'une ETC. La stratégie de comparaison par pivot propose un classement continu correct des systèmes, même si ces systèmes ne sont pas évalués aux mêmes époques. Le cadre propose également une comparaison des performances axée sur une époque spécifique. L'analyse des performances attendues projette les performances d'un système évalué dans une collection de tests différente vers l'époque d'intérêt, ce qui signifie que tous les systèmes comparés sont sur la même échelle de performances. Dans le même objectif, la stratégie des granularités propose de comparer les performances du système sur une échelle normalisée en utilisant des granularités significatives.

Il y a encore des possibilités d'amélioration du cadre d'évaluation continue, avec des expériences supplémentaires axées sur certains choix qui nécessitent des investigations approfondies. Bien qu'elles soient justifiées, la validité des hypothèses, la définition des granularités et l'estimation des projections restent à explorer. Néanmoins, nous démontrons que, lorsque des paramètres et des choix adéquats sont utilisés, notre cadre est capable de décrire avec précision les comportements des systèmes de recherche d'information sur des collections évolutives simulées et acquises.

B.3 Conclusion

Dans ce travail, nous abordons le problème de l'évaluation continue des performances des systèmes de recherche d'information lorsque les systèmes et les collections de tests évoluent. Cela se traduit par la comparaison de différents systèmes évalués dans différentes collections de tests. Nous dressons un aperçu des éléments et des limites de l'évaluation actuelle en recherche d'information. Une exigence essentielle est l'utilisation d'une collection de tests commune pour comparer les performances des systèmes, ce qui rend impossible l'application du paradigme d'évaluation classique à une évaluation continue.

Pour construire une évaluation continue des systèmes d'information, nous résumons notre contribution en deux points principaux :

- Premièrement, nous définissons une collection de tests évolutive, comprenant une formalisation, une simulation et un cadre d'acquisition.
- Deuxièmement, nous proposons un cadre d'évaluation continue des deltas de résultats capable de comparer des systèmes évalués dans une collection de tests évolutive en utilisant les $\mathcal{R}\Delta_M$.

Les contributions décrites fournissent les ressources et les méthodologies nécessaires pour mettre en œuvre une évaluation continue des systèmes de recherche d'information en utilisant des comparaisons de $\mathcal{R}_{se}\Delta_M$ entre les systèmes sur différentes époques.

Nos perspectives de recherche se concentrent sur la création de cadres d'évaluation continue des systèmes de recherche d'information explicables et interactifs, et leur intégration dans les futurs systèmes de recherche d'information adaptables. Les mesures de $\mathcal{R}_{se}\Delta_M$, utilisant notre cadre d'évaluation continue, décrivent quand les performances d'un système s'améliorent en fonction des différentes époques de la collection de tests. Nous proposons d'inclure la diversité de connaissances ($\mathcal{K}\Delta$) dans le schéma d'explicabilité pour la performance du système dans le cadre de l'évaluation continue. Nous nous intéressons notamment à expliquer les raisons de l'échec du système avec une analyse axée sur les caractéristiques de la collection de tests. Le cadre d'évaluation continue explicatif soutient une évaluation interactive, où le système de recherche d'information peut adapter son modèle et apporter des modifications en fonction des changements dans l'environnement. Cela permet une nouvelle génération de systèmes de recherche d'information capables d'apprendre comment les changements du Web (mesurés par la diversité de connaissances) impactent leurs performances (mesurées par les deltas de résultats), en intégrant un processus d'adaptabilité de la recherche d'information dans le cycle de production.

Publications

[Pub1] Gabriela Gonzalez-Saez, Petra Galuščáková, Romain Deveaud, Lorraine Goeuriot, and Philippe Mulhem. Exploratory visualization tool for the continuous evaluation of information retrieval systems. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023.

[Pub2] Petra Galuščáková, Romain Deveaud, Gabriela Gonzalez-Saez, Philippe Mulhem, Lorraine Goeuriot, Florina Piroi, and Martin Popel. Longeval-retrieval: French-english dynamic test collection for continuous web search evaluation. 2023.

[Pub3] Gabriela Gonzalez-Saez, Alaa El-Ebshihy, Tobias Fink, Petra Galuščáková, Florina Piroi, David Iommi, Lorraine Goeuriot, and Philippe Mulhem. Towards result delta prediction based on knowledge deltas for continuous ir evaluation. In *Proceedings of the workshop QPP++ 2023: Query Performance Prediction and Its Evaluation in New Tasks, co-located with The 45th European Conference on Information Retrieval (ECIR)*, pages 20–24, 2023.

[Pub4] Rabab Alkhalifa, Iman Bilal, Hsuvas Borkakoty, Jose Camacho-Collados, Romain Deveaud, Alaa El-Ebshihy, Luis Espinosa-Anke, Gabriela Gonzalez-Saez, Petra Galuščáková, Lorraine Goeuriot, et al. Longeval: Longitudinal evaluation of model performance at clef 2023. In *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part III*, pages 499–505. Springer, 2023.

[Pub5] Alaa El-Ebshihy, Tobias Fink, Gabriela Gonzalez-Saez, Florina Piroi, Petra Galuščáková, David Iommi, Lorraine Goeuriot, and Philippe Mulhem. Predicting retrieval performance changes in evolving evaluation environments. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, 2023.

[Pub6] Gabriela Gonzalez-Saez. Continuous Result Delta Evaluation of IR Systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval - Doctoral Consortium (SIGIR '22), July 11–15, 2022, Madrid, Spain.*

[Pub7] Gabriela Gonzalez-Saez, Philippe Mulhem, and Lorraine Goeuriot. Multi-element protocol for IR experiments stability: Application to the TREC-COVID test collection. In *CIRCLE 2022, July 4-7, 2022 Samatan, Gers, France*.

[Pub8] Gabriela Gonzalez-Saez, Philippe Mulhem, and Lorraine Goeuriot. Towards the evaluation of information retrieval systems on evolving datasets with pivot systems. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, 2021.

[Pub9] Sáez, Gabriela González, Lorraine Goeuriot, and Philippe Mulhem. Addressing different evaluation environments for information retrieval through pivot systems. 2021.

[Pub10] Hanna Suominen, Lorraine Goeuriot, Liadh Kelly, Laura Alonso Alemany, Elias Bassani, Nicola Brew-Sam, Viviana Cotik, Darío Filippo, Gabriela González-Sáez, Franco Luque, Philippe Mulhem, Gabriella Pasi, Roland Roller, Sandaru Seneviratne, Rishabh Upadhyay, Jorge Vivaldi, Marco Viviani, and Chenchen Xu. Overview of the clef ehealth evaluation lab 2021. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, 2021.

[Pub11] L. Goeuriot, H. Suominen, Liadh Kelly, L. Alemany, Nicola Brew-Sam, Viviana Cotik, D. Filippo, Gabriela González Sáez, Franco Luque, P. Mulhem, G. Pasi, Roland Roller, Sandaru Seneviratne, J. Vivaldi, Marco Viviani, and Chenchen Xu. Clef ehealth evaluation lab 2021. In *ECIR*, 2021.

[Pub12] Lorraine Goeuriot, Gabriella Pasi, Hanna Suominen, Elias Bassani, Nicola Brew-Sam, Gabriela Gonzalez-Saez, Rishabh Gyanendra Upadhyay, Liadh Kelly, Philippe Mulhem, Sandaru Seneviratne, Marco Viviani, and Chenchen Xu. Consumer health search at clef ehealth 2021. In *CLEF 2021 Evaluation Labs and Workshop: Online Working Notes, CEUR-WS*, 2021.

[Pub13] Lorraine Goeuriot, Hanna Suominen, Liadh Kelly, Antonio Miranda-Escalada, Martin Krallinger, Zhengyang Liu, Gabriella Pasi, Saez, Gabriela Gonzalez, Marco Viviani, and Chenchen Xu. Overview of the clef ehealth evaluation lab 2020. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 255–271. Springer, 2020.

[Pub14] L. Goeuriot, H. Suominen, Liadh Kelly, Zhengyang Liu, G. Pasi, Gabriela González Sáez, Marco Viviani, and Chenchen Xu. Overview of the clef ehealth 2020 task 2: Consumer health search with ad hoc and spoken queries. In *CLEF*, 2020.

[Pub15] P. Mulhem, Gabriela González Sáez, Aidan Mannion, D. Schwab, and Jibril Frej. Lig-health at adhoc and spoken ir consumer health search: expanding queries using umls and fasttext. In *CLEF*, 2020.

Bibliography

[1] Scipy 1.0: fundamental algorithms for scientific computing in python.

[2] T. Abdulghani, M. A. Najar, R. Belaroussi, J. Mothe, M. Ryzhov, and S. Samoskaite. Browsing information retrieval system results. In J. Mothe, P. Cellier, and A. Ligozat, editors, *COnférence en Recherche d'Informations et Applications - CORIA 2018, 15th French Information Retrieval Conference, Rennes, France, May 16-18, 2018. Proceedings*. ARIA, 2018.

[3] M. Alaofi, L. Gallagher, D. McKay, L. L. Saling, M. Sanderson, F. Scholer, D. Spina, and R. W. White. Where do queries come from? In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2850–2862, 2022.

[4] R. Alkhalifa, I. Bilal, H. Borkakoty, J. Camacho-Collados, R. Deveaud, A. El-Ebshihy, L. Espinosa-Anke, G. Gonzalez-Saez, P. Galuščáková, L. Goeuriot, et al. Longeval: Longitudinal evaluation of model performance at clef 2023. In *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part III*, pages 499–505. Springer, 2023.

[5] J. Allan, D. Harman, E. Kanoulas, D. Li, C. Van Gysel, and E. M. Voorhees. Trec 2017 common core track overview. In *TREC*, 2017.

[6] G. Amati. *Information Retrieval*, pages 1970–1975. Springer New York, New York, NY, 2018.

[7] N. Asadi, D. Metzler, T. Elsayed, and J. Lin. Pseudo test collections for learning web search ranking functions. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1073–1082, 2011.

[8] J. A. Aslam and E. Yilmaz. A geometric interpretation and analysis of r-precision. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 664–671, 2005.

[9] J. A. Aslam and E. Yilmaz. Inferring document relevance from incomplete information. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 633–642, 2007.

[10] J. A. Aslam, E. Yilmaz, and V. Pavlu. A geometric interpretation of r-precision and its correlation with average precision. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 573–574, 2005.

[11] G. Awad, K. Curtis, A. A. Butt, J. Fiscus, A. Godil, Y. Lee, A. Delgado, J. Zhang, E. Godard, B. Chocot, L. Diduch, J. Liu, Y. Graham, , and G. Quénot. An overview on the evaluated video retrieval tasks at trecvid 2022. In *Proceedings of TRECVID 2022*. NIST, USA, 2022.

[12] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[13] P. Bailey, A. Moffat, F. Scholer, and P. Thomas. Uqv100: A test collection with query variability. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 725–728, 2016.

[14] S. M. Beitzel, E. C. Jensen, and O. Frieder. Average precision histogram., 2009.

[15] A. Berto, S. Mizzaro, and S. Robertson. On using fewer topics in information retrieval evaluations. In *Proceedings of the 2013 Conference on the Theory of Information Retrieval*, pages 30–37, 2013.

[16] P. Borlund. The concept of relevance in IR. *Journal of the American Society for information Science and Technology*, 54(10):913–925, 2003.

[17] G. E. Box, W. H. Hunter, S. Hunter, et al. *Statistics for experimenters*, volume 664. John Wiley and sons New York, 1978.

[18] T. Breuer, N. Ferro, N. Fuhr, M. Maistro, T. Sakai, P. Schaer, and I. Soboroff. How to measure the reproducibility of system-oriented ir experiments. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 349–358, 2020.

[19] C. Buckley, D. Dimmick, I. Soboroff, and E. Voorhees. Bias and the limits of pooling for large collections. *Information retrieval*, 10:491–508, 2007.

[20] C. Buckley and E. Voorhees. Retrieval evaluation with incomplete information, acm sigir 2004 proceedings, pp. 25–32, 2004. *Google Scholar Google Scholar Digital Library Digital Library*.

[21] C. Buckley and J. A. Walz. The trec-8 query track. In *TREC*, 1999.

[22] T. Buntinx-Krieg, J. Caravaglio, R. Domozych, and R. P. Dellavalle. Dermatology on reddit: elucidating trends in dermatologic communications on the world wide web. *Dermatology online journal*, 23(7), 2017.

[23] R. Campos, G. Dias, A. M. Jorge, and A. Jatowt. Survey of temporal information retrieval and related applications. *ACM Computing Surveys (CSUR)*, 47(2):1–41, 2014.

[24] B. Carterette. But is it statistically significant? statistical significance in ir research, 1995-2014. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1125–1128, 2017.

[25] B. Carterette, A. Bah, and M. Zengin. Dynamic test collections for retrieval evaluation. In *Proceedings of the 2015 international conference on the theory of information retrieval*, pages 91–100, 2015.

[26] B. Carterette, E. Kanoulas, V. Pavlu, and H. Fang. Reusable test collections through experimental design. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 547–554, 2010.

[27] M. Cattelan and S. Mizzaro. Ir evaluation without a common set of topics. In *Conference on the Theory of Information Retrieval*, pages 342–345. Springer, 2009.

[28] D. Chakravorti, K. Law, J. Gemmell, and D. Raicu. Detecting and characterizing trends in online mental health discussions. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 697–706. IEEE, 2018.

[29] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 621–630, 2009.

[30] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 1–10, New York, NY, USA, Apr. 2009. Association for Computing Machinery.

[31] Y.-S. Chiang, Y.-Z. Liu, C.-F. Tsai, J.-K. Lou, M.-F. Tsai, and C.-J. Wang. Recdelta: An interactive dashboard on top-k recommendation for cross-model evaluation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3224–3228, 2022.

[32] H. Choi and H. Varian. Predicting the present with google trends. *Economic record*, 88:2–9, 2012.

[33] A. Chuklin, I. Markov, and M. d. Rijke. Click models for web search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 7(3):1–115, July 2015.

[34] A. Chuklin, P. Serdyukov, and M. De Rijke. Click model-based information retrieval metrics. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 493–502, 2013.

[35] C. L. Clarke, N. Craswell, and I. Soboroff. Overview of the trec 2009 web track. Technical report, WATERLOO UNIV (ONTARIO), 2009.

[36] C. Cleverdon. The cranfield tests on index language devices. In *Aslib proceedings*, volume 19, pages 173–194. MCB UP Ltd, 1967.

[37] P. Clough and M. Sanderson. Evaluating the performance of information retrieval systems using test collections. 2013.

[38] E. Cosijn and P. Ingwersen. Dimensions of relevance. *Information Processing & Management*, 36(4):533–550, 2000.

[39] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, E. M. Voorhees, and I. Soboroff. Trec deep learning track: Reusable test collections in the large data regime. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 2369–2375, 2021.

[40] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 87–94, New York, NY, USA, Feb. 2008. Association for Computing Machinery.

[41] A. El-Ebshihy, T. Fink, G. González-Sáez, F. Piroi, P. Galuščáková, D. Iommi, L. Goeuriot, and P. Mulhem. Predicting retrieval performance changes in evolving evaluation environments. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, 2023.

[42] K. M. Elbedweihy, S. N. Wrigley, P. Clough, and F. Ciravegna. An overview of semantic search evaluation initiatives. *Journal of Web Semantics*, 30:82–105, 2015.

[43] N. Ferro, Y. Kim, and M. Sanderson. Using collection shards to study retrieval performance effect sizes. *ACM Transactions on Information Systems (TOIS)*, 37(3):1–40, 2019.

[44] N. Ferro and M. Sanderson. Sub-corpora impact on system effectiveness. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 901–904, 2017.

[45] N. Ferro and M. Sanderson. Improving the accuracy of system performance estimation by using shards. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 805–814, 2019.

[46] N. Ferro and G. Silvello. A general linear mixed models approach to study system component effects. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 25–34, 2016.

[47] N. Ferro and G. Silvello. Toward an anatomy of ir system component performances. *Journal of the Association for Information Science and Technology*, 69(2):187–200, 2018.

[48] J. Frej, D. Schwab, and J.-P. Chevallet. WIKIR: A python toolkit for building a large-scale Wikipedia-based English information retrieval dataset. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1926–1933, Marseille, France, May 2020. European Language Resources Association.

[49] N. Fuhr. Some common mistakes in ir evaluation, and how they can be avoided. In *Acm sigir forum*, volume 51, pages 32–41. ACM New York, NY, USA, 2018.

[50] P. Galuščáková, R. Deveaud, G. Gonzalez-Saez, P. Mulhem, L. Goeuriot, F. Piroi, and M. Popel. Longeval-retrieval: French-english dynamic test collection for continuous web search evaluation. *arXiv preprint arXiv:2303.03229*, 2023.

[51] L. Goeuriot, J. Mothe, P. Mulhem, F. Murtagh, and E. SanJuan. Overview of the clef 2016 cultural micro-blog contextualization workshop. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 7th International Conference of the CLEF Association, CLEF 2016, Évora, Portugal, September 5-8, 2016, Proceedings 7*, pages 371–378. Springer, 2016.

[52] L. Goeuriot, H. Suominen, L. Kelly, A. Miranda-Escalada, M. Krallinger, Z. Liu, G. Pasi, G. G. Saez, M. Viviani, and C. Xu. Overview of the clef ehealth evaluation lab 2020. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 255–271. Springer, 2020.

[53] D. Gomes and M. J. Silva. Modelling information persistence on the web. In *Proceedings of the 6th international conference on Web engineering*, pages 193–200, 2006.

[54] G. González-Sáez. Continuous result delta evaluation of ir systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3493–3493, 2022.

[55] G. Gonzalez-Saez, A. El-Ebshihy, T. Fink, P. Galuščáková, F. Piroi, D. Iommi, L. Goeuriot, and P. Mulhem. Towards result delta prediction based on knowledge deltas for continuous ir evaluation. In *Proceedings of the workshop QPP++ 2023: Query Performance Prediction and Its Evaluation in New Tasks, co-located with The 45th European Conference on Information Retrieval (ECIR)*, pages 20–24, 2023.

[56] G. Gonzalez-Saez, P. Galuščáková, R. Deveaud, L. Goeuriot, and P. Mulhem. Exploratory visualization tool for the continuous evaluation of information retrieval systems. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023.

[57] G. Gonzalez-Saez, P. Mulhem, and L. Goeuriot. Towards the evaluation of information retrieval systems on evolving datasets with pivot systems. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, 2021.

[58] Q. Guo and E. Agichtein. Beyond dwell time: estimating document relevance from cursor movements and other post-click searcher behavior. In *Proceedings of the 21st international conference on World Wide Web*, pages 569–578, 2012.

[59] S. H. Hashemi, C. L. Clarke, A. Dean-Hall, J. Kamps, J. Kiseleva, et al. An easter egg hunting approach to test collection building in dynamic domains. In *EVIA@ NTCIR*, 2016.

[60] C. Hauff, D. Hiemstra, F. De Jong, and L. Azzopardi. Relying on topic subsets for system ranking estimation. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1859–1862, 2009.

[61] K. Hofmann, L. Li, and F. Radlinski. Online evaluation for information retrieval. *Foundations and Trends in Information Retrieval*, 10(1):1–117, 2016.

[62] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[63] E. C. Jensen, S. M. Beitzel, A. Chowdhury, and O. Frieder. Repeatable evaluation of search services in dynamic environments. *ACM Transactions on Information Systems (TOIS)*, 26(1):1–es, 2007.

[64] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Acm Sigir Forum*, volume 51, pages 4–11. Acm New York, NY, USA, 2017.

[65] K. M. Jose, T. Nguyen, S. MacAvaney, J. Dalton, and A. Yates. Diffir: Exploring differences in ranking models' behavior. In *Proceedings of the 44th International*

ACM SIGIR Conference on Research and Development in Information Retrieval, pages 2595–2599, 2021.

[66] J. Keller, T. Breuer, and P. Schaer. Evaluating temporal persistence using replicability measures. In *Conference and Labs of the Evaluation (CLEF) Working Notes*, CEUR Workshop Proceedings (CEUR-WS.org), 2023.

[67] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne. Controlled experiments on the web: survey and practical guide. *Data mining and knowledge discovery*, 18:140–181, 2009.

[68] C. D. Manning. *An introduction to information retrieval*. Cambridge university press, 2009.

[69] R. McGill, J. W. Tukey, and W. A. Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, 1978.

[70] W. McKinney et al. pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9, 2011.

[71] M. Men, S. S. Fung, and E. Tsui. What's trending: a review of social media in ophthalmology. *Current Opinion in Ophthalmology*, 32(4):324–330, 2021.

[72] W. Mendenhall, D. Wackerly, and R. Scheaffer. Hypothesis testing. *Mathematical Statistics with Applications*, pages 427–491, 1990.

[73] R. Mihalcea and P. Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411, 2004.

[74] S. Mizzaro. Relevance: The whole history. *Journal of the American society for information science*, 48(9):810–832, 1997.

[75] S. Mizzaro. How many relevances in information retrieval? *Interacting with computers*, 10(3):303–320, 1998.

[76] S. Mizzaro. The good, the bad, the difficult, and the easy: something wrong with information retrieval evaluation? In *European Conference on Information Retrieval*, pages 642–646. Springer, 2008.

[77] S. Mizzaro and S. Robertson. Hits hits trec: exploring ir evaluation results with network analysis. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 479–486, 2007.

[78] A. Moffat and J. Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems (TOIS)*, 27(1):1–27, 2008.

[79] P. Mulhem, G. G. Saez, A. Mannion, D. Schwab, and J. Frej. Lig-health at adhoc and spoken ir consumer health search: expanding queries using umls and fasttext. In *CLEF 2020*, 2020.

[80] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. Ms marco: A human generated machine reading comprehension dataset. *choice*, 2640:660, 2016.

[81] A. Ntoulas, J. Cho, and C. Olston. What's new on the web? the evolution of the web from a search engine perspective. In *Proceedings of the 13th international conference on World Wide Web*, pages 1–12, 2004.

[82] H. Oosterhuis. Learning from user interactions with rankings: A unification of the field. *SIGIR Forum*, 54(2), aug 2021.

[83] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[84] G. Penha, A. Câmara, and C. Hauff. Evaluating the robustness of retrieval pipelines with query variation generators. In *Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part I*, pages 397–412. Springer, 2022.

[85] M. Popel, M. Tomkova, J. Tomek, Łukasz Kaiser, J. Uszkoreit, O. Bojar, and Z. Žabokrtský. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. *Nature Communications*, 11(4381):1–15, 2020.

[86] S. Priya, R. Sequeira, J. Chandra, and S. K. Dandapat. Where should one get news updates: Twitter or reddit. *Online Social Networks and Media*, 9:17–29, 2019.

[87] D. R. Radev, H. Qi, H. Wu, and W. Fan. Evaluating web-based question answering systems. In *LREC*. Citeseer, 2002.

[88] L. Rashidi, J. Zobel, and A. Moffat. Evaluating the Predictivity of IR Experiments. *SIGIR 2021 - Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1667–1671, 2021.

[89] K. Roberts, D. Demner-Fushman, E. M. Voorhees, S. Bedrick, and W. R. Hersh. Overview of the trec 2022 clinical trials track. In *Proceedings of the Thirty-First Text REtrieval Conference (TREC 2022)*, 2022.

[90] S. Robertson. A new interpretation of average precision. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 689–690, 2008.

[91] S. Robertson. On the history of evaluation in ir. *Journal of Information Science*, 34(4):439–456, 2008.

[92] S. E. Robertson. The methodology of information retrieval experiment. *Information retrieval experiment*, 1:9–31, 1981.

[93] S. E. Robertson and E. Kanoulas. On per-topic variance in ir evaluation. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 891–900, 2012.

[94] K. Roitero, J. S. Culpepper, M. Sanderson, F. Scholer, and S. Mizzaro. Fewer topics? a million topics? both?! on topics subsets in test collections. *Information Retrieval Journal*, 23(1):49–85, 2020.

[95] M. E. Rorvig. Retrieval performance and visual dispersion of query sets. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of The Eighth Text REtrieval Conference, TREC 1999, Gaithersburg, Maryland, USA, November 17-19, 1999*, volume 500-246 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 1999.

[96] G. G. Sáez, L. Goeuriot, and P. Mulhem. Addressing different evaluation environments for information retrieval through pivot systems. 2021.

[97] T. Sakai. Evaluating evaluation metrics based on the bootstrap. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 525–532, 2006.

[98] T. Sakai. Statistical reform in information retrieval? In *ACM SIGIR Forum*, volume 48, pages 3–12. ACM New York, NY, USA, 2014.

[99] T. Sakai. A simple and effective approach to score standardisation. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, pages 95–104, 2016.

[100] T. Sakai. How to run an evaluation task: With a primary focus on ad hoc information retrieval. *Information Retrieval Evaluation in a Changing World: Lessons Learned from 20 Years of CLEF*, pages 71–102, 2019.

[101] T. Sakai. On fuhr's guideline for ir evaluation. In *ACM SIGIR Forum*, volume 54, pages 1–8. ACM New York, NY, USA, 2021.

[102] T. Sakai. On the two-sample randomisation test for IR evaluation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1980–1984, 2021.

[103] T. Sakai, N. Ferro, I. Soboroff, Z. Zeng, P. Xiao, and M. Maistro. Overview of the ntcir-14 centre task. In *Proceedings of the 14th NTCIR Conference on Evaluation of Information Access Technologies. Tokyo, Japan*, 2019.

[104] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American society for information science*, 41(4):288–297, 1990.

[105] T. Samar, A. Bellogín, and A. P. de Vries. The strange case of reproducibility versus representativeness in contextual suggestion test collections. *Information Retrieval Journal*, 19(3):230–255, 2016.

[106] M. Sanderson. *Test collection based evaluation of information retrieval systems*. Now Publishers Inc, 2010.

[107] M. Sanderson and M. Braschler. Best practices for test collection creation and information retrieval system evaluation. *TrebleCLEF Project: http://www. trebleclef. eu*, 2009.

[108] M. Sanderson, A. Turpin, Y. Zhang, and F. Scholer. Differences in effectiveness across sub-collections. In *Proceedings of CIKM'2012*, pages 1965–1969, 2012.

[109] M. Sanderson and J. Zobel. Information retrieval system evaluation: effort, sensitivity, and reliability. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 162–169, 2005.

[110] L. Schamber. Relevance and information behavior. *Annual review of information science and technology (ARIST)*, 29:3–48, 1994.

[111] F. Scholer, D. Kelly, and B. Carterette. Information retrieval evaluation using test collections. *Information Retrieval Journal*, 19(3):225–229, 2016.

[112] S. Seabold and J. Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[113] M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 623–632, 2007.

[114] M. D. Smucker, J. Allan, and B. Carterette. Agreement among statistical significance tests for information retrieval evaluation at varying sample sizes. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 630–631, 2009.

[115] I. Soboroff. Dynamic test collections: measuring search effectiveness on the live web. In *Proceedings of SIGIR'2006*, pages 276–283, 2006.

[116] I. Soboroff. Meta-analysis for retrieval experiments involving multiple test collections. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 713–722, 2018.

[117] I. Soboroff, S. Huang, and D. Harman. Trec 2020 news track overview. In *TREC*, 2020.

[118] M. E. Stevens. Problems of evaluation. In *Automatic Indexing: a State-of-the-art Report*, chapter 7, pages 143–164. U.S. National Bureau of Standards, Washington D. C., 1965.

[119] H. Suominen, L. Goeuriot, L. Kelly, L. A. Alemany, E. Bassani, N. Brew-Sam, V. Cotik, D. Filippo, G. González-Sáez, F. Luque, et al. Overview of the clef ehealth evaluation lab 2021. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 12th International Conference of the CLEF Association, CLEF 2021, Virtual Event, September 21–24, 2021, Proceedings 12*, pages 308–323. Springer, 2021.

[120] M. Tamannaee, N. Arabzadeh, and E. Bagheri. Vis-trec: A system for the in-depth analysis of trec_eval results. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 2181–2184, New York, NY, USA, 2020. Association for Computing Machinery.

[121] A. Tonon, G. Demartini, and P. Cudré-Mauroux. Pooling-based continuous evaluation of information retrieval systems. *Information Retrieval Journal*, 18(5):445–472, 2015.

[122] Gabriela Gonzalez-Saez, P. Mulhem, and L. Goeuriot. Multi-element protocol for IR experiments stability: Application to the TREC-COVID test collection. In *CIRCLE 2022, July 4-7, 2022 Samatan, Gers, France.*

[123] J. Urbano, H. Lima, and A. Hanjalic. A new perspective on score standardization. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1061–1064, 2019.

[124] J. Urbano and T. Nagler. Stochastic simulation of test collections: Evaluation scores. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 695–704, 2018.

[125] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.

[126] A.-M. Vercoustre, J. Pehcevski, and V. Naumovski. Topic difficulty prediction in entity ranking. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 280–291. Springer, 2008.

[127] E. Voorhees, T. Alam, S. Bedrick, D. Demner-Fushman, W. R. Hersh, K. Lo, K. Roberts, I. Soboroff, and L. L. Wang. Trec-covid: constructing a pandemic information retrieval test collection. In *ACM SIGIR Forum*, volume 54(1), pages 1–12. ACM New York, NY, USA, 2021.

[128] E. M. Voorhees. Evaluation by highly relevant documents. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 74–82, 2001.

[129] E. M. Voorhees. The philosophy of information retrieval evaluation. In *Evaluation of Cross-Language Information Retrieval Systems: Second Workshop of the Cross-Language Evaluation Forum, CLEF 2001 Darmstadt, Germany, September 3–4, 2001 Revised Papers 2*, pages 355–370. Springer, 2002.

[130] E. M. Voorhees. The trec robust retrieval track. In *ACM SIGIR Forum*, volume 39, pages 11–20. ACM New York, NY, USA, 2005.

[131] E. M. Voorhees. The trec 2005 robust track. In *ACM SIGIR Forum*, volume 40, pages 41–48. ACM New York, NY, USA, 2006.

[132] E. M. Voorhees et al. Overview of the trec 2005 robust retrieval track. In *Trec*, 2005.

[133] E. M. Voorhees, D. Samarov, and I. Soboroff. Using replicates in information retrieval evaluation. *ACM Transactions on Information Systems (TOIS)*, 36(2):1–21, 2017.

[134] L. L. Wang, K. Lo, Y. Chandrasekhar, R. Reas, J. Yang, D. Eide, K. Funk, R. Kinney, Z. Liu, W. Merrill, et al. Cord-19: The covid-19 open research dataset. *ArXiv*, 2020.

[135] W. Webber, A. Moffat, and J. Zobel. Score standardization for robust comparison of retrieval systems. In *Proc. 12th Australasian Document Computing Symposium*, pages 1–8, 2007.

[136] W. Webber, A. Moffat, and J. Zobel. Score standardization for inter-collection comparison of retrieval systems. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 51–58, 2008.

[137] W. Webber and L. A. Park. Score adjustment for correction of pooling bias. In *Proceedings of SIGIR'2009*, pages 444–451, 2009.

[138] F. Zampieri, K. Roitero, J. S. Culpepper, O. Kurland, and S. Mizzaro. On topic difficulty in ir evaluation: The effect of systems, corpora, and system components. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 909–912, 2019.

[139] Y. Zhang, W. Chen, D. Wang, and Q. Yang. User-click modeling for understanding and predicting search-behavior. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1388–1396, 2011.

[140] J. Zobel. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 307–314, 1998.

[141] J. Zobel and L. Rashidi. Corpus Bootstrapping for Assessment of the Properties of Effectiveness Measures. *International Conference on Information and Knowledge Management, Proceedings*, pages 1933–1952, 2020.