FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Building a Domain-Specific Search Engine that Explores Football-Related Search Patterns

João Paulo Madureira Damas

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Sérgio Nunes

Co-Supervisor: José Devezas

July 13, 2020

Building a Domain-Specific Search Engine that Explores Football-Related Search Patterns

João Paulo Madureira Damas

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. Maria Cristina de Carvalho Alves Ribeiro, PhD
External Examiner: Prof. Bruno Emanuel da Graça Martins, PhD
Supervisor: Prof. Sérgio Sobral Nunes, PhD

July 13, 2020

Abstract

Storing and finding information is not a recent problem. Information Retrieval (IR) appeared as a research area that sought techniques to better serve user information needs. Nowadays, search engines are the main focus of IR and a key component in accessing online information. In this work, we targeted the search engine of a Portuguese sports-related website, zerozero.pt, which presented an opportunity for search result quality improvement.

We started by conducting a Query Log Analysis process to understand how users interacted with the system. For it, we made use of a search log with two weeks worth of interactions during March 2020. Following steps from literature, we preprocessed the dataset to eliminate bot activity, observing how little their presence was felt. We then split analysis into three recurrent perspectives (term, query and session level). Furthermore, we introduced a rarer viewpoint with click level analysis, proposing a set of existing metrics that could be replicated by future researchers for benchmarking, as well as a query satisfaction metric of our own, derived from the TF-IDF metric. Results showed how some typical Web search patterns remained present, such as short queries and sessions. Inspection of specific periods demonstrated how the COVID-19 pandemic negatively affected search volume, while also changing the behavior of remaining users, as well as how football games introduced localized spikes that spanned little outside the event's timeframe. Click analysis revealed how the current system failed in answering longer queries and how quickly shorter queries lead to ambiguous result access patterns when they weren't previous popular searches.

We then proposed a Federated Search solution, where each collection would represent an entity type searchable within the system, using Apache Solr for querying each resource and a Python Flask server to merge results. Additionally, we extended previous work on individual search term weighing that used past search terms for documents as a relevance substitute. These weights were calculated from a term's frequency. We instead used them as a complement, in an attempt at better predicting results for future, similar searches. To incorporate term weights, we experimented with four strategies, resultant of combining two binary variables: integration with default relevance (linear scaling or linear combination) and search term frequency (raw value or log-smoothed).

To evaluate our solution, we extracted two query sets from search logs: one with frequently submitted queries, and another with ambiguous result access patterns, according to entropy. We resorted to click-through information as a relevance proxy. By acknowledging its threats, we tried to mitigate them by evaluating under distinct IR metrics, including MRR, MAP, NDCG and Success. Moreover, we also measured Spearman rank correlation coefficients to test similarities between produced rankings and ideal orderings according to user access patterns. Results showed consistency across metrics in both query sets. Previous search terms were key to obtaining higher metric values, with those that used pure search term frequency performing best. As for integration with default relevance, there was a slight advantage for linear scaling. Compared to the zerozero baseline, our best strategies were able to match results on frequent queries and improve answering quality on ambiguous queries (up to ~6p.p better performance on most metrics), where we had previously identified issues.

Keywords: information retrieval, query log analysis, search terms indexing, domain-specific search, apache solr, website search

Resumo

Armazenar e encontrar informação não são problemas recentes. Recuperação de Informação (RI) surgiu como uma área de investigação que procurou técnicas para melhor servir as necessidades de informação dos utilizadores. Atualmente, os motores de busca são o grande foco da RI e uma parte fulcral no acesso a informação online. Neste trabalho, abordamos o motor de busca de um *website* português de desporto, zerozero.pt. Este apresentava uma oportunidade para a melhoria da qualidade dos resultados das suas pesquisas.

Começamos por conduzir um processo de Análise de Registos de Interrogações para perceber como é que os utilizadores interagiam com o sistema. Para tal, fizemos uso de um ficheiro com registos de interações ao longo de 2 semanas durante Março 2020. Seguindo os passos da literatura revista, pré-processamos os dados para eliminar atividade de origem não humana, observando que a sua presença era pouco sentida. De seguida, partimos a análise em 3 perspetivas comuns (termo, interrogação e sessão). Para além disso, introduzimos um ponto de vista mais raro com a análise de cliques, propondo um conjunto de métricas existentes, facilmente replicáveis por futuros investigadores para comparação, bem como uma métrica de satisfação de uma interrogação nossa, derivada da métrica TF-IDF. Os resultados obtidos mostraram como alguns padrões típicos de pesquisa Web permaneceram presentes, tal como pesquisas e sessões curtas. A inspeção de períodos específicos demonstraram como a pandemia COVID-19 afetou negativamente o volume de pesquisas, para além de mudar o comportamento dos restantes utilizadores, bem como jogos de futebol introduziram picos de pesquisa localizados que não se prolongaram para além o período em torno do mesmo. A análise de cliques revelou como o sistema atual falhava em responder a interrogações mais longas e como as mais curtas, rapidamente, levavam a padrões de acesso a resultados ambíguos, exceto quando se tratavam de interrogações previamente populares.

Após esta análise, propusemos uma solução de Pesquisa Federada, onde cada coleção representava um tipo de entidade pesquisável no sistema, usando Solr para interrogar cada recurso, e um servidor Python Flask para combinar resultados. Além disso, extendemos trabalho prévio na indexação pesada de termos de pesquisa individuais que usava pesquisas passadas para um documento como um equivalente para a sua revelância. Estes pesos eram calculados com base na frequência dos termos usados. Em vez disso, nós usamo-los como um complemento, numa tentativa de melhor prever resultados futuros para pesquisas semelhantes. Por forma a incorporar estes termos pesados, experimentamos 4 estratégias, resultantes da combinação de 2 variáveis binárias: integração com a relevância por omissão (combinação linear ou escalamento) e frequência do termo de pesquisa (valor puro ou suavizado via logaritmo).

Para avaliar a solução, extraímos dois conjuntos de interrogações dos registos recolhidos: um com interrogações frequentes, e outro com padrões ambíguos de acesso a resultados, de acordo com entropia. Recorremos a cliques como uma aproximação a relevância. Reconhecendo os respetivos perigos, tentamos mitigá-los, avaliando sob diferentes métricas de RI, incluindo MRR, MAP, NDCG e Success. Para além disso, também medimos o coeficiente de correlação de Spearman para testar similaridades entre as ordenações produzidas e outras ideais de acordo com

padrões de acesso a resultados dos utilizadores. Os resultados obtidos mostraram consistência nos valores entre métricas para ambos os conjuntos. Termos de pesquisa passados foram chave na obtenção de valores mais elevados nas métricas, com as estratégias que usaram a frequência pura do termo a obterem melhor performance. Quanto a integração com relevância por omissão, houve uma pequena vantagem para o escalamento. Comparativamente ao sistema atual, as nossas melhor estratégias foram capazes de igualar resultados em interrogações frequentes e melhorar a qualidade das respostas nas mais ambíguas (até mais ~6p.p em algumas métricas), onde tínhamos, previamente, identificado problemas.

Palavras-chave: recuperação de informação, análise de registos de interrogações, indexação de termos de pesquisa, pesquisa em domínio específico, apache solr, pesquisa em website

Acknowledgements

The last 5 years have truly been some experience. When I first arrived at FEUP, in September 2015, I was a lost 18 year old kid that somewhat liked computers and, therefore, ended up studying Informatics, just because. Now, at the end, I feel that I have grown, both as a person and as a professional, something that could have not been achieved without the experiences shared with other people who I must thank.

Thank you to both my supervisors, who guided me in this last chapter: professor Sérgio Nunes, for his valuable experience and constantly trying to push me further, and José Devezas, who, even though already had a lot of work with the last stages of his own PhD dissertation, still took on the challenge and found the time to provide new, challenging perspectives and ideas.

Thank you to all the friends I have made in these years: for being there in the best and worst of times, for pushing through when deadlines were crushing us, for helping me comprehend myself better, for the great discussions on the most random of subjects, and just being there, even if I not always understood it. Wherever we go from here, know that you helped me get to where I am. Also know that Eclipse still sucks, soldiers aren't left behind, and the Tottenham Hotspur stadium still awaits our visit.

Thank you to my family: my biggest pillar and inspiration, and always my driving force. You have given me everything and I wish that I can somehow do the same for you. A special word to my mother: I envy your strength. This work, and everything I've accomplished, is most dedicated to you. Thank you.

João Damas

"Here's the thing, kid. We don't get to choose how we start in this life.
Real 'greatness' is what you do with the hand you're dealt."

Victor Sullivan, *Uncharted 3: Drake's Deception*

Contents

List of Figures

List of Tables

List of Listings

Abbreviations

WWW	*World Wide Web*
IR	Information Retrieval
QLA	Query Log Analysis
TLA	Transaction Log Analysis
DIR	Distributed Information Retrieval
COVID-19	Coronavirus Disease 2019
AP	Average Precision
MAP	Mean Average Precision
RR	Reciprocal Ranking
MRR	Mean Reciprocal Ranking
NDCG	Normalized Discounted Cumulative Gain
EQS	Entropy Query Set
FQS	Frequent Query Set

Chapter 1

Introduction

The task of storing and later finding information is an age-old problem. With the appearance of computers and digital storage, stored information grew and finding it became harder. A necessity for new and increasingly automated techniques for information indexing and ranking led to the appearance of the area of Information Retrieval (IR). With the rise of the World Wide Web, information spread almost immediately and people could access it from anywhere. Finding it was not always easy and usually involved browsing a directory or querying a search engine. As search is IR's focus, attention towards it grew instantly [8, Chapter 1].

Search engines take a user query, which is their representation of some information need, match documents in one or more internal indexes, score them, and finally return an ordered list of documents deemed relevant for that query. Relevance is a key concept in IR and has led to the adoption of different strategies in different stages of the retrieval process in order to enhance returned results. A famous example is the inclusion of page link analysis by Google [13]: due to the Web's network effect, pages that are pointed to more often can usually be considered of higher quality and authority than those which do not.

Search engines are also present beyond the open Web paradigm, and also appear as solutions for more specific search environments and tasks. For instance, in many enterprises, there are specialized engines to search the different resources internal to the company. Other examples include online search of a specific domain's resources (i.e., domain-specific search), or even offline, local search (e.g., Desktop Search).

1.1 Context

The present work concerns zerozero.pt[1], a website that started in 2003 initially focused on delivering all types of football-related information at national and international level. More recently, it expanded to international domains and added other sports to their coverage, such as volleyball, hockey and handball. This work focused on football for mainly the Portuguese domain, where most of the website's activity is present. This system started with a single relational database,

[1] https://www.zerozero.pt

which still holds today, although recent work has demonstrated a desire to adopt dedicated information retrieval technologies and tools. Currently, users query the website only through direct keyword matching. This may not be enough anymore to ensure good results, especially for rarer queries that lie on the long tail portion of query frequency.

Despite these concerns, not much is objectively known about the current situation: how well exactly is the current setup answering users' information needs? How do users search the platform? These were important questions that had to be answered before any attempt at altering the retrieval process. The benefits of answering both questions would serve different groups of people: with the search pattern analysis, the owners of the platform would have a more systematic overview of their search system and its quality. Moreover, enhancements in the IR process would allow its users to experience better results for their searches.

1.2 Goals

Our work had two main objectives. First, we wanted to describe the current system in three ways: what information were we indexing, how well did it answer users' information needs, and what were zerozero.pt's users' search patterns and behaviors. For the latter two, we made use of query logs provided by the zerozero.pt team. These logs are a source of real system interaction and their collection does not interfere with the user's session, therefore presenting valuable information about potential areas of improvement [39]. The provided logs contained information on what result a user clicked and what their position was in the results list, therefore we could drive our quality analysis around this click information. The remaining log information would also be used to characterize user search behavior, which helped understand where the current system might be lacking. From this analysis and findings on related work, we would assess what indexing or retrieval strategies, beyond their traditional concepts, could be applied to enhance search results.

1.3 Document structure

The rest of the document is structured as follows: in Chapter 2, we survey Query Log Analysis, their techniques, methodologies, benefits and limitations, and previous studies performed. Chapter 3 presents some background on IR, search engines, their evolution into different, more specific areas, and more recent work that explores new strategies in the indexing and retrieval processes. Chapter 4 presents the problem at hand and where it lies within the areas previously identified and studied, as well as our solution proposal. In Chapter 5, we perform a QLA process on the collected logs and analyze search patterns from users in the sports domain and how they relate to other types of searchers. Chapter 6 contains a small description of the indexed collections and details the implementation of the prototype search engine that is our solution, with Chapter 7 presenting the solution's evaluation process, including datasets, metrics and results discussion. Finally, a reflection on the work developed and possible future expansions, in Chapter 8, concludes the document.

Chapter 2

Query Log Analysis

2.1 Overview

People use search systems in different ways. More or less experienced, with well-defined ideas about what they are looking for or simply in an attempt at discovering potentially useful information, their use cases vary and it is important to know how they are using the system to see if their expectations are being met. With the rise of the WWW, this importance grew, as web search engines ought to serve a larger user base with distinct information needs and knowledge about the system. Transaction logs are a common resource used for this user behavior analysis. Peters defines them as *"electronically recorded interactions between on-line information retrieval systems and the persons who search for the information found in those systems"* [61]. This definition can be expanded to search logs on the Web: the information retrieval systems are the search engines, while users can not only be people but also robots acting on their behalf or independently [39].

As we've mentioned in the previous chapter, these logs constitute sources of real system interaction, and their collection does not interfere with the user's session, therefore presenting valuable information about potential areas of improvement [39]. Non-interference is an important concept, as there are other methods, namely observation, that explicitly capture users' needs and behavior. This observation can cause unease, which can refrain them from acting normally, thus questioning the legitimacy of the captured data.

Given this data, we can analyze it by Transaction Log Analysis (TLA), which is the study of the previously recorded interactions between the users and the system [61]. Query Log Analysis (QLA) fits into TLA as a specific kind of transaction, targeting transactions that represented searches (queries) to the engine. With this analysis, it is possible to obtain a comprehensive overview of the users' search behavior, such as common searches, one time or recurrent trends, seasonality spikes and others. Section 2.4 presents an overview of QLA case studies, showing the different perspectives and types of analysis adopted in this process.

In spite of being an important research area, with implications in website optimization strategies and Web and Intranet search engine design (as their quality affects people's desire to use

them), often there is no practical use for the large amount of data collected, leaving out potentially crucial business information [41].

2.2 Limitations

Despite the advantages pointed out previously, TLA has often been criticized for some of its drawbacks. Even after White et al. [91] and Agosti et al. [3] suggested that the implicit feedback captured by these logs can be as good as explicit feedback. One main criticism is that transaction logs only capture a user's description of their information need and not the need itself, limiting analysis, as it is not possible to know what was the original intent or the user's perception of how the search engine is responding [39]. Traditionally, IP addresses were used to uniquely identify visitors, however, with protocols that automatically assign and reuse IP addresses such as Dynamic Host Configuration Protocol (DHCP), this became unfeasible, and new approaches moved to the use of cookies alongside IP addresses [42]. These approaches, though they mitigate the identification problem, are not fail-proof: common-access computers (e.g., at libraries, schools, labs) and spyware constitute challenges that may invalidate the assumption that a computer is associated to one person only.

Expressing information needs is also not a trivial task. Some studies found that, even with good domain knowledge, users were unable to express their needs through the available query syntax, throwing random words together connected by boolean operators in hopes of getting good results [78].

Another aspect pointed out is that, since the collection is mainly done server-side, many important interactions done client-side (e.g., clicking the "Back" button) are invisible to the collected logs. On the other hand, session identification can sometimes be challenging. To solve this issue, some applications were developed to mitigate this lack of information. Jansen proposed one such application that could be used with transaction logs across different IR studies [39], which can log interactions with the browser toolbar, among other components.

While having client-side data is a valuable contribution, there is a trade-off associated with privacy. The legitimacy of this collection must then be carefully addressed before it is done. Confidentiality is also a factor that researchers must take into account and that often prevents search logs publication, making it hard to replicate research in this area [3]. Even with anonymization techniques, searches can sometimes be traced back to real people. One example is the AOL search log leak, that allowed tracing searches back to the original users [9]. Thus, public search logs are rare and those that are available are often obsolete.

2.3 Steps

All QLA pipelines may define their own methods to tailor the analysis to their use case. However, a common core that is followed can be identified. According to Jansen [39], it is constituted by three steps: *Collection*, which deals with selecting what interaction data will be recorded, in what

format and what is the time period surrounding the log, *Preparation*, which is concerned with data organization, cleaning and refinement, and *Analysis*, where, for example, exploratory techniques are used to extract knowledge from the raw information. The next subsections describe each step at finer detail.

2.3.1 Collection

The first step is assessing what information is necessary to collect from users' interactions with the search engine for the desired analysis to be made, that is, what will characterize an entry in the query log resultant of the collection process. Though researchers are free to choose what information they think better suits their needs, some fields are common to almost all captures:

- User Identifier - An identifier that is thought to uniquely identify a user. Usually, this is an IP address, despite the limitations pointed previously;
- Date/Timestamp - A temporal mark symbolizing when the search was made;
- Query - The term or terms submitted to the search engine by the user.

These fields are what help define user sessions, which is described in the next subsections. Other fields are often found in studies as well. Analysis of AltaVista [69] and SAPO [64] search logs included user agent information, with the former also including search modifiers and optional advanced operators. Chau et al. [21] also collect request methods and response statuses, as well as referrer URLs.

The time period is also a factor to be considered. The two mentioned studies cover a period of six weeks and approximately five and a half months, respectively. In a comparative study of nine different large scale transaction logs between 1997 and 2002, Jansen found that these time periods ranged from a single day to several months [41].

2.3.2 Preparation

After the data is collected, it needs to be properly organized, cleaned and refined. According to the needs initially defined, data storage might be done in different formats. To query structured information, relational databases might be used. Jansen, in his framework proposal, describes a generic schema capable of storing transaction logs with the most commonly captured fields [39]. In their studies, Wang et al. [86] and Chau et al. [21] also adopt this strategy. More recently, Mahmood et al. [54] concluded that NoSQL databases are also a viable storage option to perform scalable log analysis. If, during the collection period, some attention was dedicated to ensuring workable data formats, organizing the data in systems such as databases might not be necessary. In our opinion, the additional overhead of setting up a database should be considered, as in some cases it might just be an unnecessary additional layer of complexity.

The next step is to filter out data that is considered irrelevant or from unwanted sources which may negatively impact results obtained after analysis. The most common example is robot

searches, i.e., searches made by machines (which can be advertising bots or other types of bots with malicious intent). There is no guaranteed way to capture all non-human queries [69], though there are some well-known strategies. Usually, a threshold is defined on the number of queries per session: sessions with a higher number of queries than it are deemed suspicious and discarded. Silverstein et al. [69] use a value of 101 queries/session, a value fifty times over their calculated average, allowing a high enough cut-off to prevent biased results, then eliminating sessions and respective queries that surpass it. The definition of session assumes, once again, a relevant position. Sessions are generally defined by grouping queries with the same User Identifier within a specified time frame, which is the session cut-off. Estimating this value is not-trivial, and existing studies use cut-off values from 5 minutes [69] to 30 minutes [21, 80, 82], which is the approximate maximum value reported by Jansen [42]. Other strategies include setting thresholds on the number of terms per query: Yi et al. [99] discard queries with over one hundred terms. If there is enough information, one could go even further. In a study conducted by Ribeiro on a Portuguese web search engine [64], there was user agent information in the logs. Using regular expression matching and a predefined known bot agent list, he was able to filter out around 30% of all entries which were non-human. Chau et al. [21], also using this technique, identified over half (approximately 56%) of all queries as having a non-human origin.

2.3.3 Analysis

Finally, in the analysis step, the preprocessed data is analyzed and more thoroughly described using well-defined metrics and statistics. With them, it is possible not only to infer knowledge from the data being analyzed but also to compare with other similar studies to assess if findings are in accordance to previous work. If a database approach similar to the one described above was adopted, this analysis can consist of a set of SQL queries that can be tuned to maximize performance, as done by Jansen [39]. One downside is, of course, that one would be limited by the capabilities of the querying language.

To facilitate comparison between different studies, a framework was proposed by Jansen et al. [40] that decomposes data analysis at three levels corresponding to different granularities: term level, query level and session level.

A term is a sequence of characters delimited by a separator, usually a whitespace character. At this level, term frequencies are measured to assess term popularity in the data collected [69]. Further analysis includes term combination analysis and term distribution analysis (e.g., check if it follows a Zipf distribution as reported by Spink et al. [75]). Though term combination is usually done at term pair level, some studies [21] include analysis on frequency for combinations of up to 6 distinct terms.

A query represents a sequence of terms, once again delimited by a known separator, that is submitted to the search engine. Common statistics include query frequency and length, number of unique queries, frequency and depth of advanced operator usage. It is also common to distinguish different types of query, grouping analysis by different types: initial, identical/repeated and modified queries (see Chau et al. [21] for an example). An initial query represents the first query

Table 2.1: QLA analyzed studies.

Engine Type	Study	Time Period	#Days	#Records
Open Web	Altavista [69]	Aug. - Sep. 1998	43	933,208,259
	Excite [94, 74]	Dec. 1999	1	1,025,910
	AOL [10, 11]	Dec. 2003	~7	~100,000,000s
	AOL [10]	Sep. 2004 - Feb. 2005	~180	~1,000,000,000s
	SAPO [64]	Jan. - Jul. 2010	165	45,413,607
Domain-specific	Univ. Tennessee [86]	May 1997 - 2001	~1460	541,920
	BIBE [78]	2002	1	Unknown
	PsycInfo [99]	Feb. 2002	28	28,651
	ABC-Clio [99]	Feb. 2002	28	256,377
	Utah government [21]	Mar. - Aug. 2003	168	1,115,388
	INDURE [31]	Oct. 2010 - Jan. 2011	90	14,503

in a session. Subsequent queries are classified as identical/repeated (the same query has been sent before by the searcher) or modified (a new query that has not been sent by the user before). Some studies define more detailed query types, such as re-finding queries, which are queries that are not necessarily repeated but have the intention of (re)finding previously retrieved results [82]. Further work also includes query categorization into topics or categories [11, 86, 99]. These topics must usually be predefined manually, though, by using Machine Learning techniques, automatic topic discovery is likely to become a more frequent approach.

A session is defined by a sequence of queries within a delimited time span and is typically associated with some information need. At this level, researchers are interested in assessing how long sessions are and how many queries are performed per session. Other interesting kinds of analysis include query chain analysis to, for example, evaluate how users reformulate queries with bad results to achieve their information needs [42].

2.4 QLA Studies

There are a number of QLA studies available, mostly focused on all-purpose Web search engines. However, here we also present studies that target other types of platforms, e.g., website search [86, 21], as our target platform (zerozero.pt) is not an all-purpose Web search engine, but an engine built on top of a local database. Table 2.1 presents an overview of the collected studies.

The collected studies cover a range of distinct areas and time periods, allowing the inference of behavior patterns from different kinds of users throughout time. This is not an extensive collection of QLA studies, but rather a representative sample that will allow future comparison.

The AltaVista [69], Excite [94, 74], AOL [11, 10] and SAPO [64] studies target all-purpose search engines and, therefore, have an available number of queries superior to all other studies. The

last one, in particular, analyzes searches made on a Portuguese search engine. While it is expected that there is some foreign activity in zerozero.pt, most of the activity should be from Portuguese users, hence this study provides an interesting comparison benchmark. Beitzel et al. [11, 10] performed two analyzes on AOL logs, one on a broader time range, and other focused on hourly variations. The study by Wang et al. [86], focusing on queries submitted to an academic search engine, aimed at describing how search behavior changed, presenting the largest period covered in our selection. Tang et al. [78] set up a controlled experiment that had knowledgeable people in botanics try answering questions in a domain-specific search engine. Field wise, aside from the three components identified in Section 2.3.1, the AltaVista study collects information on the result screen (i.e., request search result range), user-specified filters and submission information (e.g., usage of advanced operators). For the work of Wang et al. [86], there is information on the number of hits (results) for the query submitted. In their Utah state government study, Chau et al. [21] also possess client and server IP addresses, as well as request information (status), user agent (browser) and referrer URLs.

In terms of preparation, there are some differences in the decisions made to produce analyzable data. Both studies of the University of Tennessee by Wang et al. [86] and the Utah state government website by Chau et al. [21] stored log information using relational databases, as proposed by Jansen's framework [39]. Others do not discriminate any further storage processing, so we assume they worked directly with the log files (e.g., Ribeiro [64] used a set of bash scripts based on UNIX commands). Distinguishing between queries with human origin and those machine-generated (e.g., robots or web crawlers) is sometimes ignored in studies, which may result in biased results. The AltaVista study eliminates sessions with a number of queries superior to a defined threshold of 101, while the Utah state government study utilizes the user agent (browser) field to match pre-known artificial agents, as does the SAPO study. In their academic search engine analysis, Wang et al. [86] report that, in a first phase, they separate "unusual queries from text queries", though it is not clear how this procedure was carried out. Session definition also differs between studies, which leads to differences in obtained results. In the AltaVista study, sessions have a 5 minute cut-off, though 30 minutes was the most commonly used value. Some studies had no session information (i.e., no way of uniquely identifying users) and, therefore, skip this phase. Out of all domain-specific search engine studies collected, only the Utah state government and the INDURE studies have session information and define such concept. Though the latter does not indicate a delimiter value, the former conforms with previous work and also uses a 30 minute cut-off.

Concerning term level analysis, studies arrive at different conclusions. The Excite study reports a unique term percentage of 61.6%, a value that was still stable in an analysis of logs further in time on the same engine [74], while the SAPO studies arrives at a value of only 3% for unique term values. One reason that could help explain this discrepancy is the difference during the pre-processing of each study: while Ribeiro had a thorough pipeline to clean and refine the dataset for analysis, removing close to 70% of the original queries due to their non-human origin, the Excite study is not so rigorous. In their analysis, Spink et al. [74] state that these values come as a result of the presence of terms associated with "personal names, spelling errors, non-English terms, and

[...] URLs". Yi et al. [99] reported 9% and 14% of unique terms for the history and psychology databases, respectively. For Chau et al. [21], this value was of about 4.5%. As for Tang et al. [78], the percentage of unique terms was close to 26%, a similar value to Wang et al. [86], who also report that only around 6.7% of all unique terms were used in all years that covered their research, which corroborates that users change throughout time and so do their needs. This means that engines serve more and more information needs, which demand new ways of expressing them. Some studies also report on term frequency distribution and the possible presence of the Zipf law. This law states that a term's frequency is inversely proportional to their rank in the frequency table (e.g., the most popular term appears twice as many times as the second most popular term) and, while there is no clear explanation as to why this happens, it is a pattern that has been repeatedly observed and used to help decide improvements on search engines design, such as caching frequent terms [2]. Wang et al. [86] analyzed the presence of this law and found that there was a negative linear relation for all terms and an approximate quadratic relation when only considering unique terms and that these relations held up for all the years considered, eliminating the possibility of a coincidence. Finally, Ribeiro [64] showed that, by caching the top 2% most frequent terms, it was possible to answer 90% of all queries that contained cached terms. Though there is no Zipf distribution analysis in this study, this conclusion leads us to believe that term distribution in those logs might have also followed this law.

Regarding query level analysis, a first look is given to the percentage of unique queries. Wang et al. [86] found that 72.6% of all queries only occurred once, yet this number drops to around 20% for Fang et al. [31], which resembles what was achieved by Ribeiro [64]. Though the total number of queries is not given, Yi et al. [99] state that the uniquely identified queries constitute approximately one-fourth of the total queries. When looking at query length, results appear to be very similar. The AltaVista study reported an average of 2.35 terms per query. This number rises slightly to 2.4 terms in the Excite study, a value that saw some stability in some of the earliest research on this engine, though it later raises as well [74]. This phenomenon is also seen in AOL studies, where the value changes from 2.2 [11] to 2.7 [10] in less than two years. This may indicate a tendency of increasing query length, at least in the same engine: as people become more knowledgeable and used to an engine, they will know how to best query it to achieve better results. In the INDURE study, queries average only 1.96 terms, though authors point out that expert searchers are more willing to explore other advanced search capabilities, which we will see next. Ribeiro [64] got a value of 2.31 terms, while Yi et al. [99] found that users querying history and psychology databases averaged 3.42 and 3.16 terms, respectively, a higher value than any other found. When talking about query type frequency, there isn't a consistent classification system adopted throughout existing work. In AltaVista [69] and Excite [74] studies, there is only the identification of modified queries, i.e., subsequent, unseen before queries by the same user. Results show modified query percentages of 20.4% and 39.6%, respectively. In the Univ. of Tennessee study, authors measure repeated queries ratio, with 12.4% being identified as such. Ribeiro [64] identifies queries either as initial, identical (both with the meaning as presented previously), modified (query for the same information need with some terms changed) and new (a

query with completely new terms, symbolizing a new information need), with initial and repeated queries presenting similar frequency and representing about two-thirds of all queries. When modifying queries, this study found that searchers usually didn't change the original query too much, as one term difference was the most common case. This means that users are willing to stick to their original need expression. Wang et al. [86] analyzed consecutive queries to assess how users learned in what they called a "mental model of the Web". Like in SAPO, consecutive queries usually differed by the addition or removal of terms, more specifically one at a time. This goes against findings in the AltaVista study, where users tended to totally change the query or modify some of its terms primarily. Fang et al. [31] found that expert searchers tend to use more common queries. From this, we can infer that modifications, in this context, are rarer.

Often, these modifications are a result of adding or removing boolean operators [86]. Silverstein et al. [69] describe that around 20% of queries made use of these operators, while this value drops to 5% in Excite studies [74]. The SAPO study arrives at a value of 1.5%, which seems to indicate that users, overall, are unaware of these operators, or at least their usage, when searching. One syntax more commonly used are phrase queries (i.e., terms enclosed by double quotes), which perform exact matches. Chau et al. [21] reported that, while only 3.4% of queries made use of boolean operators, 29% were phrase queries.

Some studies also report on query frequency distribution. With it, researchers assess how frequent are the top queries and, in a way, how diverse are the users' information needs (though keep in mind the same query can be used for different information needs, e.g., the query [nails] might be directed towards human nails, as in manicure, or I might be searching for the best nail type to hang up a frame on the wall). The AltaVista study arrives at around 1% for their top 10, i.e., the top 10 queries represent only one-hundredth of the total queries, which may be lower than expected. Excite studies take a different perspective and analyze the coverage of their top 100 terms, concluding that 19.3% of all term usages correspond to one of those more popular [74].

Query categorization, along with temporal topical trend analysis (i.e., seasonality) is a common aspect to most studies, even from studies that were not on all-purpose Web search engines. As stated previously, a manual categorization is the most frequent way of assessing query topics and is usually performed with a subset of all queries. For this, Spink et al. define a set of eleven general categories [74]. In the SAPO study, the main category was *Computers or Internet*, with close to 27% of queries, followed by *People, Places or Things* (about a 4 percentage point difference). Older studies, like the Excite ones [74], also rank these two high, though the top spot goes to *Commerce, Travel, Employment or Economy*. The lower percentages of the *Computers or Internet* shows how search behavior has evolved. As their search engines are specific to a given domain, to classify psychology queries, Yi et al. [99] use a classification system from the American Psychology Association and find that the most frequent category is *Psychological and Physical Disorders*. Indeed, people tend to use the Web more and more to understand illness symptoms, even if at the expense of erroneous conclusions [90]. In terms of temporal analysis, both AOL studies [11, 10] provide insights on this perspective, with the former on an hourly basis, while the latter spans a broader period of time (months). In them, the authors point out the

stability of some trends, even with the considerable query volume change. Their findings include how different trends are associated with specific times of a day: *Personal Finance* queries saw a rise in the morning period (7-10 am), while *Porn* related searches peak during nighttime. When expanding to several months, categories tended to stabilize, except some that were most prone to seasonal spikes, namely *Holidays* (e.g., Halloween, Christmas), *Shopping* (season change, which is associated with change in clothing), *Sports* (end season playoffs in baseball and football) and *Government* (2004 Presidential election). In their study, Wang et al. [86] also found that some categories had seasonal spikes, both in terms of queries and in the query volume for some categories. More specifically, weekends and holiday periods had a lower volume, while queries for academic career affairs grew during January and August/September (the beginning of semesters). Beitzel et al. comment that these trends, if properly analyzed and predicted, can help search engine developers evolve caching algorithms to prioritize stable query categories [10]. As an example, the author points out a query [eagles], that would normally correspond to searching for the bird, but during football season in the USA is most likely related to the American football team with the same nickname. Indeed, if caching queries based on their or their terms' frequency already allows to answer a good percentage of queries (90% in the case of SAPO, as we described earlier), the incorporation of this kind of analysis could raise the search engine's efficiency. Another approach would be helping rerouting queries to specialized search engines for a category, a concept also known as vertical search, which will be introduced in the next chapter.

Regarding the number of results (or pages) viewed, most often users do not go past the first page, though in Excite studies this number does not represent a majority (only 42.7%). Interestingly, percentages for viewing two, three or more pages are similar. This could mean that a user, given the patience to scroll through result pages, is willing to go all the way. Chau et al. [21], however, found that, while users mostly view less than two result pages (average of 1.47), the average number of documents retrieved per session is inferior to one, with close to 60% of sessions resulting in no documents viewed. Though it is easy to think that users were not satisfied by the search results, the authors point out another possibility, where users were satisfied with the title and summary presented in the results list, and simply moved on from there, leaving no further interactions logged. In our case, there isn't a visual summary that will likely satisfy the user's information need, so we wouldn't be able to replicate this hypothesis.

Finally, at session level, results show that, overall, sessions are short and users mostly do one query per session. The AltaVista studies show a 77.6% of sessions with one query only [69]. However, since their session cut-off value was of 5 minutes, the number of sessions was probably overestimated and some sessions considered distinct are actually the same [39]. On average, Excite studies reported 1.9 queries/session [74], a value similar to the one obtained by Fang et al. [31]. Chau et al. [21] got a slightly inferior value of 1.73 queries. In terms of session duration, values obtained are quite different: while Fang et al. [31] obtained a value of 18 minutes and 40 seconds, this value is only 5 minutes and 23 seconds for the SAPO study [64]. This difference is likely explained by the different session definition and cut-off values adopted by each researcher: Ribeiro used a cut-off of 30 minutes, and it appears that Fang et al. did not use a cut-off at all,

grouping sessions by the first and last query of every unique identifier.

2.5 Summary

QLA is a research area with solid work that provides a good basis in terms of process steps and benchmarks for comparison. Analysis is usually partitioned into three main levels: term, query and session, which measure patterns at different granularity levels.

Studies usually fall into one of two categories: open Web, if the engine indexes a snapshot of reachable content on the Internet, or domain-specific, in case the engine serves a smaller amount of information that is semi-manually controlled. Despite this categorization, many search behaviors tend to remain constant throughout users. Similarities include query length, which tends to be short, much like session length and duration. Though older results suggested patterns of complete query reformulation, more recent studies have shown that users refine their queries in a less extreme manner, more often no more than one term at a time. Moreover, in both scenarios, advanced query operators (e.g., boolean), when presented, aren't commonly used. Finally, both types of engines can be subject to seasonality spikes that affect search volume, both positively and negatively. There are also divergence points. A main distinction between both types of studies lies in the number of records available, as open Web engines, for being able to serve a broader set of information needs, are subject to much more activity. Other differences observed included term uniqueness, which was higher in some open Web studies and once again likely due to the larger user base and associated intents, and results pagination, with users on domain-specific engines less likely to navigate further than the first page.

Despite the existence of studies on domain-specific engines, with framing of results in the context they are inserted, we were unable to find studies focused on the sports area, more specifically football. Therefore, our analysis will be unique in this aspect and provide valuable information for researchers in the area.

Chapter 3

Search Engines

In order to understand search engines, we must understand the process of Information Retrieval, or IR. Storing and finding information is not a recent problem. The first archives, constituting the first types of libraries, can be traced back as far as 3000 BC [66]. With the growth of available information and its diversity, there was the need to develop solutions that could scale better and be prepared for this variety, thus new alternatives had to be found. It is around the 1940s that research in IR starts to gain momentum (e.g., it was in 1945 that Bush proposed the Memex [17], a prototype with several ideas that would much later be incorporated into the WWW), especially with the appearance of computers, though only in 1950 would Calvin Mooers [58] coin the term IR:

"The problem under discussion here is machine searching and retrieval of information from storage according to a specification by subject... It should not be necessary to dwell upon the importance of ***information retrieval*** *before a scientific group such as this for all of us have known frustration from the operation of our libraries - all libraries, without exception."*

IR systems were initially developed at an academic level, with libraries among the first to adopt their usage, and allowed limited searching on designated fields such as author and title [8, Chapter 1]. Though developments were made in the IR process in the following decades, a general concern arose in the academic community in the late 1980s, as the size of test document collections was much smaller than those used in commercial companies [66]. In the 1990s, with the appearance of the World Wide Web, this concern became a reality and the large scale of information on the Web forced new perspectives on the IR process. In fact, Web search was later identified as one of three types of IR systems by Manning et al. [55, Chapter 1], alongside personal information search and enterprise search. There was also a spawn of new search research areas, which span not only the Web but also the other IR system types just mentioned. They are described in detail in Section 3.5.

3.1 Overview

With the growth of information, its scale, necessity to search it and the association with IR, the terms *Information Retrieval* and *search* have often been used to refer to the same thing [55, Chapter 1]. In fact, a more recent definition of IR by Manning et al. [55, Chapter 1] helps to see why it is the case:

"Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)."

When a user accesses a search engine, they bring some information need to satisfy. There is then the need to translate this into textual keywords, corresponding to a query to be made to the system. This process is not always easy for the user: when the information need is poorly defined or just too generic, the user may end up in a cycle between browsing and searching [8, Chapter 1]. They start with either one and, as relevant results are found, the other is used to further refine the query's satisfaction. This contrasts with the data retrieval paradigm, where information follows a known structure and users query it using specialized querying languages (e.g., SQL for relational databases). The results obtained are either right or wrong, as there is no possible subjectivity adjacent to the query. In IR, the system must try and match as many documents that might be relevant as possible, while trying to keep the retrieved irrelevant document count low [8, Chapter 1]. Relevance is a key concept in IR and retrieved documents are usually ordered by a relevance score given by the system.

All put together, a search engine is a set of modules that deal with different stages of the retrieval process. Figure 3.1 presents an overview of an architecture for search engines. Though generic, it is still a good example of what are the main modules and how they interact during the retrieval process.

The documents that a user queries constitute the collection. How the documents arrive at the collection depends on what kind of search engine we are dealing with: in the case of Web search, there is a continuous crawling process, also represented in the figure. A crawler, given some base hyperlinks, follows every available path that it can find (through anchors) in order to try and cover the majority of the Web. Due to the Web's bow tie structure [8, Chapter 11], not all pages are accessible by anchors and, therefore, it is, in practice, impossible to index all of the Web.

There are then two main processes: indexing and retrieval (& ranking). We now cover these in finer detail.

3.2 Indexing

After we have a document collection, a question arises: how can we efficiently query it? While storing the full text of documents is feasible for small collections, with large volumes of data this

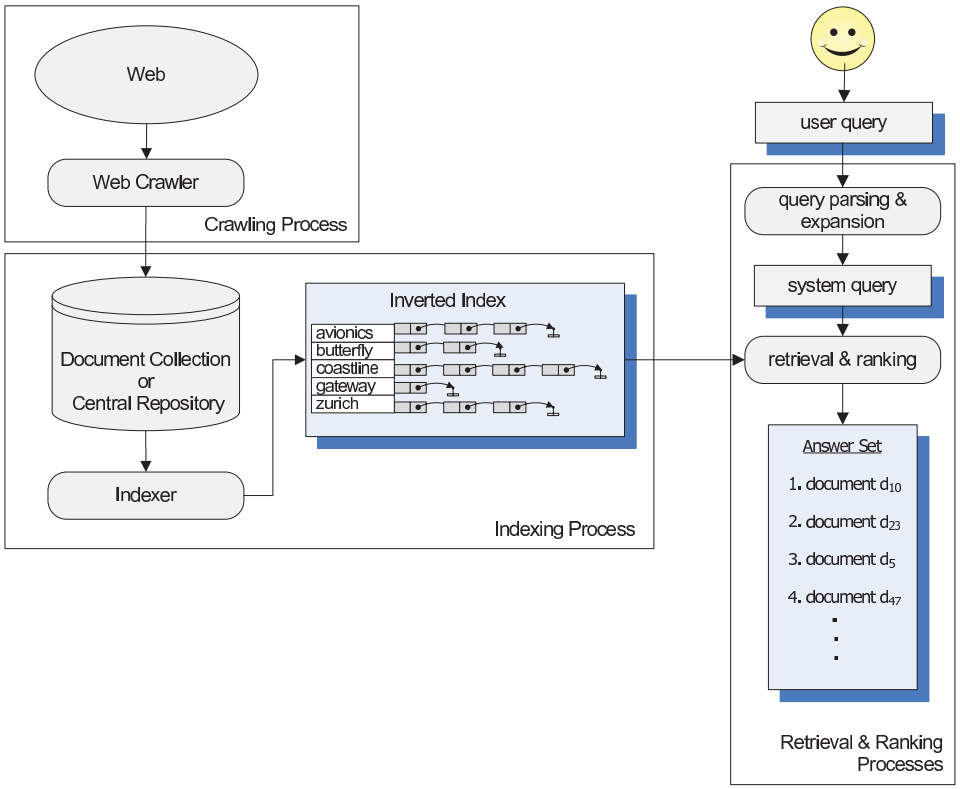


Figure 3.1: Generic search engine architecture. Retrieved from Baeza-Yates et al., *Modern Information Retrieval* [8, Chapter 1].

becomes costly. The answer is an index: much like database indexes, indexes in IR are partial representations of documents that allow querying for one or multiple terms in it.

The most common index structure used in IR systems is the inverted index. In this structure, there are two main concepts: vocabulary, which consists of all unique terms presented in the entire document collection, and postings. A postings list is a list of document identifiers where a certain term appears. Figure 3.2 shows an example of an inverted index that targets sentences from classical pieces of literature such as Shakespeare's *Hamlet*.

On the left, we have the vocabulary, formed by the different terms, and their document frequency, i.e., the number of documents in which the term appears (which effectively corresponds to the posting list's size). On the right, each term points to the document identifiers where it is present. There are variants to this simpler structure: in order to match phrase queries, for example, it is imperative to store occurrence positions in posting lists. In this case, we would have a full inverted index [8, Chapter 9].

The path from document (text) to index is not immediate. For example, we have to deal with punctuation (not all punctuation is irrelevant, e.g., the digital world gave birth to many terms that go beyond alphabetic characters, like e-mail and IP addresses [55, Chapter 2]) and casing. Other issues arise: what if we had different forms of a similar base word? If Doc 2 included the term

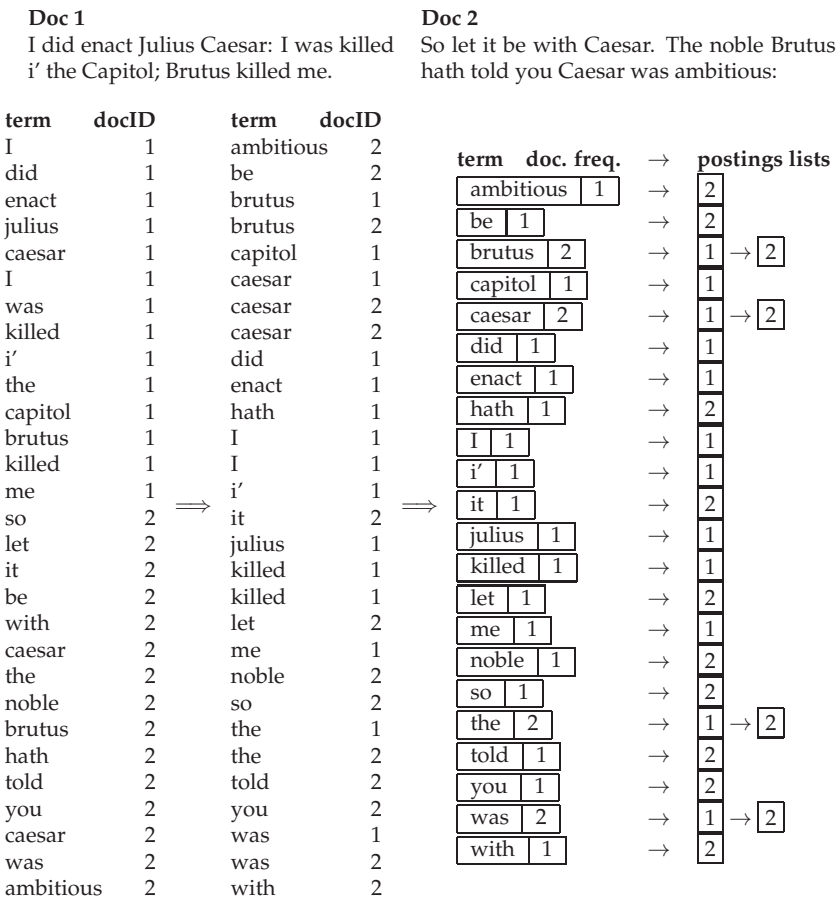


Figure 3.2: Inverted index example. Retrieved from Manning et al., *Introduction to Information Retrieval* [55, Chapter 1].

"killing", we would want to match a search for "kill" on both documents. These and other issues led to an indexing pipeline that documents go through before being really indexed. An overview of this pipeline is presented in Figure 3.3, presenting all the most common steps, from tokenization, to stopword removal and stemming.

In a first phase, there is the recognition of possible document structure. Indeed, we may be interested in separating terms in different fields (e.g., title and body) in order to be able to later force matches on specific fields. Associating this structure information results in parametric (for exact matches) or zone (for free text fields) indexes [55, Chapter 6]. The first step in text processing comes in the form of *tokenization*, that is, separating the document text into separate tokens. Word boundaries are not easy to decide: while breaking on non-alphanumeric characters is an easy solution, it goes wrong very fast (e.g., see digital concepts presented previously, or think of hyphenated words, which, in some cases, might help more if separated, and other times only make sense if kept all-together). In the next step, there is the removal of tokens that are generally of low interest for their lack of expressiveness. *Stopwords* are usually just helpers in phrase syntax and bring little discriminating power to a query (e.g., "the" appears in both documents in Figure 3.2).

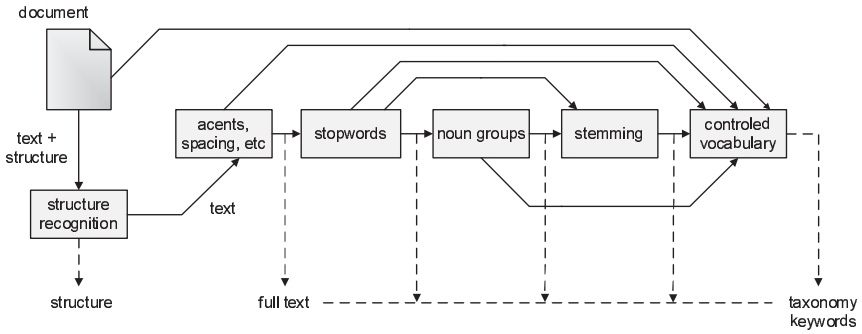


Figure 3.3: Document indexing pipeline. Retrieved from Baeza-Yates et al., *Modern Information Retrieval* [8, Chapter 1].

Finally, to deal with different variations of the same base term, *stemming* aims at cutting word suffixes in order to try and reduce them to their root form (e.g., "jumping", "jumped" and "jump" would all converge to "jump"). Although stemming algorithms achieve good performance (we highlight the Porter stemmer [62], the default stemming step included in most IR tools), they are still only heuristics. An alternative is *lemmatization*, where, with the help of a vocabulary and morphological analysis, words are reduced to their base dictionary form. However, due to its complex requirements and being a more time-consuming process, it is not as used in practice.

At the end of this pipeline, we will have an inverted index with some optimization regarding the terms indexed. There is always a trade-off in doing such processing, as it can lead to poorer results since we are not fully indexing the document for discovery [8, Chapter 9], however, this structure is a standard and produces good results.

3.3 Retrieval

Given an indexed collection and a query submitted by the user, the system must then decide what documents are relevant for that query. IR retrieval models have evolved from boolean matching to probabilistic weighting, though each has its advantages and drawbacks nonetheless. We present an overview, focusing on classic IR models, with some Web-specific developments towards the end.

Boolean model

The boolean retrieval model is the simplest model in Information Retrieval and one of the first developed. In it, queries are terms joined by boolean operators, such as AND, NOT and OR [55, Chapter 1]. There isn't a notion of a partial match: either a document fully matches the boolean expression, or it doesn't.

Let's imagine a system whose collection contains the example documents in Figure 3.2. If we sent to this system the query [Brutus AND Capitol], it would return only the first document, since it's the only that contains both terms. Mathematically, we take each document's postings

list, which can be thought of as an n-dimensional (where n is the number of documents) binary vector whose i^{th} element is 1 if the document with ID i is in the postings list, and 0 otherwise. The vectors for Brutus and Capitol would be 11 and 10, respectively. Then, to obtain documents that match the query, we apply the boolean operators with the vectors in place of their terms. In this case, 11 AND 10 yields 10, hence the first document is our query result.

Although simple to understand, as boolean logic is intuitive, some problems quickly arise. If our query was instead [Brutus OR Capitol], both documents would be returned, however, for the system, both were equally as good, which often is not the case (i.e., here we would likely prefer the first document, as it contains both terms). Moreover, if one document contained the term Brutus 100 times, while the other matched only one, this, once again, would be equal in the eyes of this model. This is another of the limitations of the boolean model: it does not rank results, only checking for (exact) matches. Even for its simplicity, we can still point out that meaningful boolean queries are hard to express: term intersections (AND) are too restrictive, yielding few results; term unions (OR) are too broad, returning potentially a lot of unwanted documents. This fine-tuning is a process that users generally are not willing to go through [55, Chapter 1].

Vector space model

We pointed out that binary weights were one of the main issues in boolean retrieval, as it only allowed for exact matches. The notion of documents and queries as vectors, though, can be generalized to solve these issues. The vector space model proposes new ways of weighting terms, both in documents and queries, which allow partial matching. Indeed, we can say that the boolean model is a particular case of the vector space model [55, Chapter 6].

To calculate a document's score, we measure their similarity with the submitted query, by calculating the dot product between both vectors. The dot product is the sum of all products of term pairs in the same index on each vector. In other words, for n-dimensional vectors $\vec{Q}$ and $\vec{D}$, their dot product can be defined as:

$$Sim(\vec{Q},\vec{D}) = \vec{Q}.\vec{D} = \sum_{t_i \in \vec{Q},\vec{D}} w_{t_{i_{\vec{Q}}}} \times w_{t_{i_{\vec{D}}}} \tag{3.1}$$

where $w_{t_{i_{\vec{Q}}}}$ is the weight of term i in vector $\vec{Q}$.

There have been many proposals on how to calculate term weights in the vector space model. Most are based on two concepts, one of which we already introduced. A term's *document frequency* (df) is the number of documents in which a term appears (not counting repetitions). *Term frequency* for a term-document pair is the number of occurrences of the term in that document. These two factors are the basis for the known TF-IDF weighting scheme [55, Chapter 6]. The reason behind these factors is that, for one, if a query term appears more often in a document, this document should be more relevant. On the other hand, a query term that appears frequently in the collection does not have a high discriminating power and, therefore, shouldn't be weighted as

much as others. For the latter, the *inverse document frequency* (idf) is used to measure a term's rarity. A term's IDF can be defined as a logarithmic-smoothed inverse ratio of a document's popularity, i.e., $log(N/df)$, where N is the collection size. A term's weight is then defined by the product of their tf and idf scores.

In order not to bias similarity scores towards longer documents, all scores should be normalized, turning the similarity measure into the cosine value of the angle formed by both vectors in the hyperspace they are defined in. Some variations exist though, for term frequency, document frequency, and normalization factors. These are presented in Figure 3.4.

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1+\log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5+\frac{0.5\times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$ (Section 6.4.4)
b (boolean)	$\begin{cases}1 & \text{if } \mathrm{tf}_{t,d}>0\\ 0 & \text{otherwise}\end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha<1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

Figure 3.4: TF-IDF variants. Retrieved from Manning et al., *Introduction to Information Retrieval* [55, Chapter 6].

One of the main limitations of the vector space model is that, like the boolean model, it assumes indexed term independence [8, Chapter 3].

Probabilistic models

Probabilistic models take another perspective on document relevance. In them, documents retrieved are ordered on the estimated relevance probability for the query submitted. This is also known as the Probability Ranking Principle [55, Chapter 11]. The first model developed was the **Binary Independence Model (BIM)**. This model was based on Bayes' probability theorem, which states that, for events A and B:

$$P(A)\times P(A|B) = P(B)\times P(B|A) \tag{3.2}$$

where A|B is defined as the event where A occurs, given that B occurs, and with $P(A)+P(B)>0$, i.e., none can be an impossible event. BIM also uses the vector representation we introduced previously with binary weights. The goal is to calculate the probabilities $P(R|\vec{D},\vec{Q})$ and $P(I|\vec{D},\vec{Q})$, that is, the probability of relevance (R) or irrelevance (I) given a document and a query. Through Bayes' theorem, we can rewrite these probabilities: $P(R|\vec{D},\vec{Q}) = \frac{P(\vec{D}|R,\vec{Q})\times P(R|\vec{Q})}{P(\vec{D}|\vec{Q})}$. The probability of a query's relevance on their own is constant and, since the model aims at ranking documents, this factor can be discarded [55, Chapter 11]. For the remainder of the formula, in order to estimate probabilities of all term incidences occurring, the model once again utilizes Bayes' theorem, assuming term independence and multiplying conditional probabilities for all terms.

The BIM inspired subsequent work on probabilistic retrieval, however, it is not used in his original form anymore for its flaws: for one, it does not account for term frequency (tf) and document frequency (df) factors, previously introduced in the vector space model and shown to have high discriminating power. On the other hand, it also does not properly take into account document size.

One known evolution of BIM is the **Okapi BM25** model, named after the first system it was implemented on [46]. Unlike BIM, Okapi BM25 takes into account the factors mentioned above and can be computed without relevance information. It is the result of empirical experiments and tuning over time: it is calculated using two previous iterations, BM11 and BM15. Since its proposal, it has been considered as the best baseline model when researching for new improvements on retrieval, substituting the classical vector space model [8, Chapter 3].

Other approaches

Other approaches have also been studied for the retrieval process. To try and use text structure information, semi-structured retrieval models such as **Proximity Nodes** and **Non Overlapping Lists** have been proposed [8, Chapter 13]. The former explores a hierarchical index structure, allowing multiple structures in the same document text. A query can then match against different hierarchies, though it only returns results from one. The latter consists of dividing the document text into non-overlapping regions and storing them in lists. Usually, this division occurs in the form of chapters or sections. There is a single inverted index whose entries are the structural components identified, with text regions associated as posting elements.

Language models have also been explored as a retrieval process. One common example is the Query Likelihood Model [55, Chapter 12]. This model also has its basis on Bayes' theorem, but takes a different perspective and ranks documents on the probability that a query would be the result of a random sample from the corresponding document. Another used method relies on the Kullback-Leibler divergence. With origins in information theory, this measure assesses how bad one model is at simulating other, and is usually used to see if a query's terms would be good enough to model a potential relevant document (i.e., the lower score, the better the document is) [55, Chapter 12]. One main disadvantage of this approach is that comparison across queries is not possible [55, Chapter 12].

More recently, with advancements in Machine Learning algorithms, **Learning to Rank** [55, Chapter 10] approaches have also been studied, where systems learn which documents are relevant for the type of query they receive.

Retrieval on the Web

On the Web, instead of text documents, we have pages as our indexing units. Web pages are unique in the sense that they do not only contain text but also can point to each other through anchors. This linking allows us to view the Web as a directed graph, where nodes represent pages and edges hyperlinks from one page to another. This has led to research in *link analysis* and the development

of algorithms that score pages based on these links. **PageRank** [13] is an algorithm that measures a page's quality by the number of quality pages that point to it. It is based on the notion that important (quality) pages are likely to be linked to from other pages. Around the same time, Kleinberg et al. [48] developed what was later called the **Hubs and Authorities** algorithm. In it, pages are assigned two scores, a Hub score and an Authority score. A page should have a higher Hub score if it points to pages with high Authority values, and these latter ones have this status if they are pointed to by high-quality Hubs. While these new analyzes and takes on the Web's link endorsement effect are interesting, they can be easily manipulated: in the case of PageRank, there could be malicious pages pointing to each other just to boost their PageRank scores. Therefore, in reality, Web search engines use hundreds of signals, making these approaches just one part of a larger calculation [55, Chapter 21].

3.4 Tools

There are a number of tools to help implement search engines. We present a general description of the top five, as of the moment of writing, from the DB-Engines' ranking [25].

Elasticsearch [29] is part of a tech stack that also can serve as a document database. It is based on the Lucene core project [4] and, being a more recent project built on a different stack, it is usually praised for its better scalability. **Splunk** [76] comes in second place, however it is more of a Big Data analytics platform. It is a commercial project owned by the company with the same name. **Solr** [70] is the last of the top three. Very similar to Elasticsearch feature-wise, as it is also based on Lucene core, it is an older project, with more support and documentation. It also has client APIs for several programming languages, is extensible through programmable plugins and comes with an administrator interface where the server can be configured and debugging can take place. **MarkLogic** [57] is similar to Splunk, as it is also a commercial product that is mostly oriented towards serving as a database. Finally, **Sphinx** [73] is a project aimed at easing querying of different sources, namely databases of different types, providing APIs also in several programming languages and supporting a complex query syntax.

In the end, due to their documentation and bigger maturity and support, we see Solr as a more appropriate tool for most use cases and chose it as the tool to implement the new search engine. This decision was also helped by the fact that the zerozero.pt team had already been experimenting with it for an upgrade of the search engine themselves.

3.5 Emergence of New Search Tasks

Information Retrieval is an area naturally oriented towards and, therefore, molded by tasks. The annual Text REtrieval Conference (TREC)[1], the largest of the area, focuses, every year, on different tracks that target and reflect the variety of unique contexts where search is present.

[1] https://trec.nist.gov/ (accessed May 2020)

While Web search is an important research area, there have been other search tasks that rose more or less at the same time as a result of more specific needs. These tasks sometimes differ in their scale compared to Web search and bring a set of new challenges, both in the indexing and retrieval processes. Some are also recognised as an independent IR system type by Manning et al. [55, Chapter 1]. We present an overview of each one, stating how they differ, when compared to the more traditional search, and the main challenges they pose.

3.5.1 Enterprise Search

Enterprise search appeared as an answer to the need of finding information on an organization's information systems. White [87] describes that 93% of executives reported a revenue loss in their organizations due to inefficient searches by their employees. Indeed, in an organizational context, a searcher is typically an employee who wants to find some business-relevant information related to their work (e.g., a financial report or some protocol).

Enterprise Search is one of three main IR system types defined by Manning et al. [55, Chapter 1]. Hawking defines it as a search on an organization with electronic textual material [37]. This includes external websites, company intranet, and any other digital resources held, such as e-mails, database records and shared documents. From here we can already see one major difference: the need to search very heterogeneous collections that differ in structure (a part of documents is still mostly unstructured text, where IR excels, whereas other documents have significant structure that may influence search) and format and storage types bring additional indexing challenges that we had not faced before. Though this also holds for other types of modern search engines, in an enterprise this effect is most felt. On the other hand, documents in an enterprise often are stored in multiple languages [89], making it harder to search for them. Another important factor is restricted access: enterprises store documents that may contain sensitive information that should not be accessible to everyone inside it, thus security is a concern when designing these search engines [14].

In fact, creating a tool that efficiently searches across these different resources often requires fine-tuned solutions for the enterprise's specific use cases. Information systems' size in enterprise search are smaller than those on Web search engines and lack the anchor hyperlink nature of Web pages [14], invalidating the usage of existing link analysis algorithms that helped Web search engines achieve more successful results. Another retrieval challenge lies within the expected user profile: while in Web search, users might be satisfied with some relevant results in the top 10, in enterprise search users can lie on the other end of the spectrum and expect the best answer at number one [89], which is the only thing they are really looking for. Adapting search results to the user's context is one of the main challenges in enterprise search [37]. In spite of this, when talking about crucial success factors for enterprise search, White [88] points out that enterprise search users should be trained and have access to other sources of advice and help, as well as referring to the requirement for the search engine to adapt to the users' inability to express their need and the consequent query refinement process that follows (comparing search to a dialogue).

3.5.2 Desktop Search

If Web Search can be characterized by its large scale, Desktop Search, also known as Personal IR [28], sits on the opposite side of this spectrum. The third type of IR system, scale-wise, identified by Manning et al. [55, Chapter 1], targets personal usage in one's computer. With Operating Systems integrating small IR systems in their software (e.g., Apple's macOS Spotlight) and e-mail clients being more and more used, this has also become an interesting research area.

Like in Enterprise Search, the main challenge in the indexing process lies within the various types of documents that people have on their personal computers. However, here, we deal with other differences, as multimedia and other file types with little text beyond the filename are likely to be more common. This brings challenges to built efficient search engines, while being nearly maintenance-free and lightweight (so it can run without disturbing the user's session) [55, Chapter 1]. Not much public research, though, is made, as public test collections are hard to obtain (e.g., for privacy and confidentiality issues). Nevertheless, some researchers point out that these systems should be adapted to the users' recent memory: people often search for specific details about documents they previously found [28, 33] and it is important to understand what kind of information they tend to recall about their documents [12].

3.5.3 Federated Search

The diversity of document types to be indexed has been one of the major challenges so far in emerging search tasks. If some research, on one hand, tries to find ways of indexing all these resources centrally, others have explored the possibilities of distributed and parallel computing applied to IR systems. Distributed Information Retrieval (DIR) [20] is a research area that explores these capabilities by isolating different document types on separate indexes and engines.

Federated Search is one of the main applications of Distributed Information Retrieval techniques. Indeed, both terms are even used interchangeably [34]. In Federated Search, there is a set of specialized search engines for the different types of documents that can be retrieved. The environment in which these collections are inserted is often categorized as cooperative or uncooperative (competitive) [5]. In a cooperative environment, all resources work together to try and bring the best results in an efficient matter. An example would be if all collections are operated by the same company (like in an enterprise). This, of course, requires extra effort in coordination between resources. In an uncooperative environment, information sources are interested in results being extracted from their collection.

In this search task, there are essentially four phases: *Resource Description*, *Resource Selection*, *Results Merging* and *Results Presentation* [20]. The first phase concerns the indexing process. Collection representations must provide information to help selection. In uncooperative environments, this involves resource sampling techniques, where a set of queries and their results allow the central system to collect statistics to describe each resource. In cooperative environments, the need for cooperation demands communication overhead. The next two phases concern the retrieval process. *Resource Selection* deals with assessing what are the relevant collections to obtain result

documents from. Selection approaches can be categorized into large document, where collection documents are merged and the scoring process is performed on these, or small document, where sample centralized indexes score resources based on the number of documents and their respective position achieved in a common ranking [22]. We describe some of the algorithms developed for these approaches in the next subsection. After resources are selected and their relevant documents retrieved, in the *Results Merging* phase there is the need to order them in a single ranking. Even if they have similarities in the retrieval process, they are not equal and, therefore, query scores for documents from different resources cannot be directly compared [5]. Hence, all scores must be normalized not only according to their collection-specific score, but also the selection score of the resource collection itself. *Results Presentation* deals with how to integrate the different results and types in the interface display (beyond a simple merged list).

3.5.4 Aggregated Search

In Federated Search, we spoke of specialized search engines for different document types. However, most of these types are usually assumed to be of easy textual representation. In Aggregated Search, the focus is not only on different document types but also on different media types. Aggregated Search can be seen as a set of technologies that help existing resources (also called verticals in this context) from highly specialized search engines to complement traditional results from IR methods. One typical example, and where Aggregated Search research is most present, is on the Web [6]. For example, when searching for a person, not only are relevant Web pages returned, but also a set of other entities, such as news, other people and images (each indexed by a specialized vertical), that the engine considers being relevant and related to the search. This happens because people might be searching for this related content instead of what the query would return by itself [49]. This heterogeneity resembles Desktop Search, as there is the need to deal with similar circumstances in it.

Aggregated Search is, then, another area where DIR techniques are applied [22]. The stages involved are similar to what we presented in Federated Search. The contrasts in each are what separates both: in Aggregated Search, the environment is always cooperative. This and the variety in media types across verticals make *Resource Description* a more challenging task than in Federated Search (Section 3.5.3). Because of this, there are no unique approaches that can search across all verticals the same way [6]. However, even before that, during *Resource (Vertical) Selection*, the goal in both is not the same: in Aggregated Search, there is no need for at least one vertical to be considered best. Since we are appending related information to main results from another engine, the interest is in finding if there is one or more verticals that indeed help to better complement the underlying information need, rather than answer the query itself. This difference is key in the way the selection works: while in Federated Search, selection is score-based, since we want to rank resources in a competitive fashion, in Aggregate Search this selection is binary, deciding only if the vertical is relevant with no interference on other verticals (unless there is some threshold set on how many can be used) [6].

While there could be other specific search tasks, we chose to focus on these as they were the most present in the literature reviewed, which includes proposals on new indexing and retrieval techniques that will be presented next.

3.6 New Strategies for New Demands

With the appearance of new search tasks and the challenges that come with them, researchers experiment with new strategies, both in the indexing and retrieval processes, not only to overcome these challenges but also achieve improvements over existing baseline models. We will now present an overview of collected and reviewed literature on new strategies for each process, their advantages and limitations, and adequacy for our case.

3.6.1 Indexing Strategies

The basic inverted index structure based on document content presents some limitations: by viewing the document as a *bag of words*, it cannot process phrase queries or just take advantage of term co-occurrence, as some terms only make sense when put together (e.g., "New" and "York" by themselves are almost meaningless, however "New York" is a common occurrence that is likely to be searched frequently). On the other hand, it does not capture alternative document representations: people might remember the document not by their content, but by some description given by another person other than the author, or even from previous searches that led to it.

These were the focus of reviewed literature. For one, to index documents by terms that are related to the document, but may not be in the document itself, serving as alternative representations; but also, to store term co-occurrence information (where co-occurrence can correspond to term pairs, trios or bigger combinations, though there is a performance trade-off associated with using bigger combinations and most work does not surpass term trios). The latter is used in conjunction with early termination techniques (e.g., [102, 98, 67]), which targets efficiency in the retrieval process by skipping score calculation for some documents and will be described more thoroughly in the next section.

3.6.1.1 Alternative document representations

Research focusing on alternative document representations targets mostly enterprise-like environments, e.g., local website search. As we've noted before, the social force that drives online content and structure is not present in these situations: on the Web, content is the voice of a public, while in Intranets it is the voice of an entity; in the former, the number of documents grows in an uncontrolled fashion, while in the latter this growth is managed by an organization.

Fagin et al. [30] suggest a set of axioms that describe Intranets: i) documents are created for simplicity, not attraction; ii) most queries have few (or even one) correct answers that do not stand out from other documents; iii) Intranets are virtually spam-free; and iv) a large portion of intranet documents are not search-friendly. In their work, they present a system that uses three

separate indices to index an Intranet's pages: one for their content, one for the title and related metadata (e.g., meta HTML tags) and one that combines all anchor text leading to that page. Anchor text is the text bound to an anchor containing a hyperlink [30]. They justify their decision based on axioms two and three: as users are looking for (nearly) unique resources, a separate title index might help isolate that page; on the other hand, the lack of spam dictates that all anchor texts are valuable alternative document representations with no malicious intents. Ding et al. [27] use a similar approach, but instead of an index dedicated to title and metadata, they construct an index based on previous searches. They reason that combining multiple sources of evidence leads to better results, as one alone "is not enough to construct a good website search engine, especially when the page is new or seldom accessed". This is an important point that resembles our work closely: we expect to have a heavy tail of pages that are less visited and, therefore, would suffer from a lack of data problem if trying to use solely log data for indexing. In their tests, they compared their proposed approach with a single index with all the combined content and concluded that the former performed better. Moreover, the log index proved most effective in retrieving some top results instead of a full relevant set. These kinds of results would likely fit better in our domain as well.

Zhou et al. [100] also use a three-fold index using anchor texts and search log entries, however, their work differs on index construction: in the anchor text index, they adopt a sliding window to capture surrounding text as well, with a limit of 80 characters. Moreover, as their logs contain referrer page information, they perform term propagation between consecutive pages visited in each user session. For this, they first extract entry terms from the referrer's anchor text to the page visited (or from the query itself if the referrer is the search results page). This procedure is repeated for all pages, and all other than the first also receive terms that are propagated from previous pages in the session (following the user's visiting path). Not all propagated terms have the same importance, as terms from pages closer to the one being processed assume higher weights. Their best results also came when linearly combining the separate indices instead of merging all terms into a single structure. Oakes et al. [59] take an extreme approach and index documents only by previous search terms. They argue that, in a system with multilingual documents and searches, this strategy allows to merge all previous searches, regardless of language, providing users with easier access to documents in a broader set of languages, as many can be present in the index (their analysis mostly focused on how query language varies within a session). Terms are weighted in a TF-IDF like approach: those that are frequently used to search for a document (term frequency) and not used often to reach others (document frequency) will have a higher weight for that term-document pair. In our case, this strategy alone wouldn't likely work, due to the heavy-tail limitation presented above.

3.6.1.2 Term co-occurrence

Regarding term co-occurrence, Schenkel et al. [67] present three index structures to incorporate term-pairs. The first has entries for each term and associated Okapi BM25 scores (i.e., does not consider term co-occurrence); the second stores, for each unordered term pair and document, a

proximity score based on previous work by Büttchel et al. [18] that combined BM25 and term proximity; the third, alongside this information, also stores independent term Okapi BM25 scores. On the latter two, sorting is made by the proximity score for the term pair. This allowed for different structural combinations to obtain several processing strategies. In their work, the authors use a combination of the first and third structures, using the third as an extension whose term scores allowed to discriminate documents with high term proximity scores early on (i.e., if a document is seen in a term's dimension, but not the term pair's, it will not appear in the other term's either). Moreover, they use static pruning, removing entries with lower scores to reduce index size. They report that this pruning affected term pair entries more, as they have lower scores more often, which is expected if calculating entries for all possible pair combinations.

Zhu et al. [102] state that term co-occurrence was usually incorporated into top-k retrieval algorithms, but this made them inefficient, thus the need to study how to enhance index structure to accommodate this information. In further work [103], they introduce three ways of indexing term-pairs: the first two are similar to other work, adopting a traditional inverted index, with term pairs as entries, and a second one that extends this by storing term proximity scores. Their third structure records extended hit-lists (hits and near-hits). As an example, the hit list for "New York" in document D could be (A9, R82, L91), symbolizing a phrase hit at position 9, a right pair term only hit at position 82 and a left pair term only hit at position 91. This offers some flexibility when compared to other structures as it offers more information than simply pre-computed scores. However, it comes at the cost of a larger index, making pruning a must. The authors enforce this by only creating entries for pairs with at least one popular term, with a term's popularity defined by their document frequency (df). However, for our case, if we only indexed popular terms, heavy tail queries, with rarer terms, would suffer even more. Yan et al. [98] showed that using a maximum distance of 3 terms is enough to build term pair indexes and describe an idea similar to the one we just spoke of, but with a slightly different implementation: for each term pair, there is a list of entries, each with a document identifier, the pair frequency in the document and a list of occurrences that store information on whether a full or partial match was recorded and in what position. They also propose the integration of this structure with existing single term indices and conclude that integrating with indices that consider document structure is most effective.

Finally, Fontoura et al. [32] use a new strategy that involves storing term co-occurrences using bitmap indices. A bitmap index stores binary occurrence information on a fixed number of columns. Here, for each term, the index stores information on co-occurrence for some other terms determined *a priori*. They approximate this term selection to a greedy optimization problem, thus easing index construction. Furthermore, they show that this strategy does not substitute, but rather complements pre-computed lists like the ones presented in previous work. They present a hybrid index building algorithm that, at each step, chooses between one or the other according to whichever maximizes the marginal gain obtained when compared to the current structure. Their results showed improvements in query performance, even for heavy-tail queries, with a growth of just 3% in index size. However, their approach only targets conjunctive boolean queries, with authors proposing extension for other query types as future work.

3.6.2 Retrieval Strategies

The traditional retrieval processes presented have some known limitations. Term proximity information not being incorporated, as we described in the previous section, is also a problem at this step. Link analysis algorithms, popular in the context of Web search, make assumptions that do not hold in other search tasks, such as searching only within a website (i.e., a graph that does not point to other pages on the Web). As Web search engines may use hundreds of signals, so do other search tasks demand several sources of evidence that need to be merged, as there can be only one final ranking. With Federated and Aggregated Search, there is the need to develop more effective algorithms to select resources (or verticals) that are relevant to queries. Moreover, even with a great algorithm, efficiency can still be a concern: how to calculate scores has led to different processing strategies. Turtle and Flood [81] categorize retrieval processes into two main strategies: Term at a Time (TAAT) loops through all documents for each term, thus making calculating document scores a gradual process. On the other hand, Document at a Time (DAAT) evaluates all query terms for one document before moving to the next one. Search engines then also demand efficient top-k solutions, as it may not be feasible to compute full scores for all documents and return good results at the same time. Most work that we cover uses DAAT strategies, as they are easier to integrate, for example, with term proximity scores and early termination techniques.

We now present all reviewed work categorized in a similar way to the one we just introduced. Some didn't fit any broader category and, thus, we decided to group them and present them first.

Xue et al. [96] recognise the lack of effectiveness of traditional link analysis algorithms on website search and propose a method that uses frequent pattern mining algorithms to extract generalized association rules to incorporate in the ranking process. The items mined for these rules correspond to elements of a taxonomy automatically built from the website's pages' structure (i.e., from the URL), as these environments often offer such structure and users tend to access resources at different hierarchical levels. They then use search logs to mine the generalized rules and linearly combine their support with traditional similarity scores. Though they report an improvement of 15% over pure keyword-based retrieval, this work seems to have some caveats: building a taxonomy for the whole website is a process that cannot be applied in our case, or at least in the same way authors constructed theirs: in zerozero.pt, URLs are flat and present little to no hierarchical structure. Furthermore, even with a possible workaround, it would be complex to determine the taxonomy of entries in search logs.

Kraft et al. [50] aim at expanding search by capturing and using search context information during the retrieval process. Their motivation lies with the ambiguity that is present in Web search, where queries are typically short but have unexpressed meaning behind them (e.g., the query [jaguar] can refer to the brand or the animal). In this work, authors consider context as a short textual description that can be represented using the vector space model, sorted by relevance. With this vector, three approaches are suggested to incorporate them into the retrieval process: *Query Rewriting (QR)* takes a predefined number of terms from the context vector and concatenates them to the user query, submitting the resulting search query to a standard search engine. For its

simplicity and help in restricting ambiguity (due to more terms being inserted), it may reduce the overall number of relevant results returned; *Ranking Bias (RB)* also takes some top terms from the context vector, but uses them as independent boosters for results (i.e., if two documents match the query but only one contains the context term(s) selected, that one will be ranked higher), with the possibility of each term having a different boosting weight; finally, *Iterative, Filtering Meta-Search (IFM)* builds many queries from the different possible combinations of the query and context terms and then has to merge and re-rank results from each individual sub-query to produce the final result set. This can be matched to a rank aggregation problem. Their results showed that even the simplest alternative, *Query Rewriting*, produced improvements over simple keyword-based retrieval. However, contextual search raises privacy concerns, even when applied in a controlled environment such as enterprises. In our case, users are not tied to any company and, therefore, this contextual information gathering is likely a challenge too difficult to overcome for a possible implementation.

3.6.2.1 Click-through Data

Click-through data is a way of obtaining implicit relevance feedback from search results. It consists of capturing the result pages that a user clicked on for some query and use them as a relevance judgment [44]. Though they cannot be used as ground-truth statements, as users can, for example, be exploring the possibility of a result being relevant, they convey information that can be useful for learning retrieval functions. Joachims [44] proposes a framework that uses Support-Vector Machines to accomplish this task and achieves improvements with a scalable solution, both query and feature-wise.

In follow-up work, Joachims et al. [45] evaluate the usage of click-through data as implicit feedback in Web search. One problem pointed out is that clicks are usually interpreted as endorsements for the destination pages, which is not always true due to the limitations just pointed. Indeed, top results often get more clicks, which can lead to a score reinforcement by learning algorithms, which in turn reinforces the top results' position and leads to them having more clicks, thus entering a positive feedback loop. This problem can be identified as *Trust Bias*, where users partially trust the search engine's relevance judgment or at least consider them most promising. Therefore, clicks are a good relative relevance source but can be too biased to be used blindly.

3.6.2.2 Link Analysis Revisited

To mitigate the lack of link endorsement effect that brought effectiveness to traditional link analysis algorithms, some research focuses on an evolution of those algorithms, especially PageRank, that work on closed environments. Xue et al. [97] propose a new way of calculating the input graph that, alongside explicit links that exist between pages, also considers implicit links that are calculated from previous access patterns using search logs. This way, it is possible to try and simulate the endorsement effect as a path from one page to another might also represent the support

that link analysis algorithms assume. These implicit patterns are extracted using two-item sequential pattern mining techniques (similar to previous work by some of the same authors we presented above [96]). For each link that is considered frequent, an edge is created on the website graph (or, if it already exists, it is updated). In the end, weights are normalized, so as not to bias results, and PageRank is applied. To integrate into the retrieval process, they use a linear combination with similarity between query and page in two ways: by combining scores or rankings from each algorithm. Their results show that the latter performed best, with an improvement of 16% over traditional PageRank.

Cui et al. [23] follow a similar approach in using page access frequency instead of inlinks and outlinks directly. They introduce the concept of a probabilistic graph, where nodes represent pages and an edge from A to B the probability of following that path. This new algorithm, that the authors call LPageRank, then calculates page relevance similarly to traditional PageRank: for each page, there is a teleportation term, similarly to the original PageRank proposal, with the remaining score coming from the LPageRank of pages that link to it and the probability of reaching this page from there. In other words, for a page A:

$$LPageRank(A) = \alpha + (1-\alpha) \sum_{B \in ascendants(A)} LPageRank(B) * P(B,A) \tag{3.3}$$

where α is a value in the range [0, 1] and P(B, A) is the probability of reaching A from B. This component is then used to linearly scale a TF-IDF retrieval model. Results show improvements over traditional PageRank, despite no improvements over domain-restricted Google search (i.e., using the *site:* modifier). Given Google's expected retrieval model complexity, though, this was to be expected.

3.6.2.3 Rank Aggregation

Recall that, in their work targeting intranet workplace search, Fagin et al. [30] use three separate indices: content, title (and metadata) and anchor text. In their retrieval process, they use a rank aggregation model. In it, all candidate documents (web pages) selected from the indices are ranked according to several independent heuristics, in a way that the final ranking minimizes the number of jumps in a document's place. Rank aggregation techniques had been the target of previous work, which showed that it is effective against spam [30]. Indeed, by testing documents against several different heuristics, both possible bias and weaknesses from a single measure can more easily be mitigated.

The authors use seven heuristics in this work: PageRank scores, page in-degree (i.e., number of pages that point to it), page discovery date (by the crawler; sometimes, a page being more recent can be used as a tie-breaker, hence probably why this choice), URL terms (like a mini-document), URL length (Fagin et al. argue that, while not suitable for absolute ranking, it may help decide authoritativeness of a page), URL depth (pages closer to the root are usually better) and a discriminator to purposely bias towards certain pages. We can see that the choice of heuristics can raise some questions, as some, for example, are highly correlated and might end up creating a bias

nonetheless (e.g., URL length + URL depth + URL terms). However, it is a modular architecture that allows the addition or removal of further heuristics. It is also a convenient aggregator in the absence of query classifiers.

Their experiments were run against both popular and heavy-tail queries, which is important as the latter, together, represent a non-disposable portion of all queries and are usually what lack better answer sets. Results showed that the best heuristics combination used at least four independent measures, corroborating the necessity for rank aggregation. It is expected that some heuristics help in retrieving more relevant documents, while others help rank relevant documents higher. This heuristics selection that balances out the benefits and limitations of all that are incorporated appears to be the key challenge were one to use this strategy.

3.6.2.4 Early Termination

Zhu et al. [103] use their proposed index structure to reduce the upper bound on unseen documents so that early termination can occur sooner. They rely on a *phrase-friendly* property that states that, if a document does not possess the phrase query, then their term proximity upper bound score can be reduced by a factor inferior to one. Their work, though, is only applicable to term pairs and would require rewriting properties and algorithms to extend for combinations of higher values.

The strategy adopted by Yan et al. [98] also works with term pairs. In a first phase, they check their index structure to see if there is an entry for a term pair. Given this list, they evaluate all documents present in it, as well as documents present in the list corresponding to the reverse pair, if applicable (i.e., evaluate posting list for (t1,t2) and (t2,t1)). From this evaluation comes a temporary top-k list. In a second phase, other documents are evaluated using single term indices until early termination can occur.

Broder et al. [15] introduces a two-phase retrieval strategy that, unlike most work, can rely on a traditional inverted index as we presented towards the beginning of the chapter. In the first phase, they calculate partial scores for documents and select those that surpass a defined threshold. In the second phase, previously selected documents are fully evaluated to produce the final ranking. Selection in the first phase is done using an extended boolean operator, WAND (Weak or Weighted AND). This operator, as well as boolean variables $X_1 ... X_n$, also receives a list of non-negative weights $w_1 ... w_n$ (one per variable) and a minimum threshold θ. The operator evaluates to true iff:

$$\sum_{i=1}^{n} X_i * w_i \geq \theta \tag{3.4}$$

As the value of θ increases up until n, the operator moves toward a logical AND (conversely, reducing up until 1 results in a logical OR). In their work, the boolean variables correspond to query terms that can appear or not in a document being evaluated, whereas their weights correspond to the upper bound score of each term's contribution to the document's score. Results obtained showed a reduction of over 90% in the number of full document evaluations, while not

affecting the quality of documents retrieved. This approach seems very modular and of easy implementation, however, there could be concerns over calculating upper bound scores or in the fine-tuning of the threshold.

Tao et al. [79] study two types of proximity measures: span-based (also called global approaches) and distance-based (local approaches). In the first category, there are two measures tested: *Span* is the length of shortest document containing all term occurrences (including repetitions); *MinCover* is similar, but is only interested in the first occurrence of each term. For the second category, *MinDist* corresponds to the smallest distance for every pair combination of unique query terms. Similar measures for average (*AvgDist*) and maximum (*MaxDist*) are also tested. To incorporate in the retrieval process, these measures are transformed using a smoothing log function $\pi(Q,D) = log(\alpha + exp(-\delta(Q,D)))$, where Q is a query, D a document and $\delta(Q,D)$ is one of the previous previous measure values. This function is then added to traditional Okapi BM25 and KL-Divergence retrieval model scores, e.g., $score(Q,D) = BM25(Q,D) + \pi(Q,D)$. Results show that *MinDist* is the most effective measure in assessing document relevance, but only after normalization is applied.

Despite all the work presented, Büttcher et al. [18] report that term proximity may not always be helpful in larger collections. They hypothesize two positive correlations: term proximity importance with average document size and term proximity with text collection size. While their experiment seems to confirm the latter, the former remained hypothetical, hence these strategies should be carefully considered and tested before usage.

3.6.2.5 Type Prediction

We use the Type Prediction designation somewhat loosely here. We are, indeed, talking about a task that assumes different names depending on the context it is applied, though all converge to the same base problem that is trying to be solved. In Federated Search, this is Resource Selection; in Aggregated Search, it is Vertical Selection; in Desktop Search, it is related to finding the right document type. What they have in common is that, from a set of independent collections, one (or more) must be chosen.

Kim et al. [47], in the context of Desktop Search, deal with document type prediction using type-specific metadata for the prediction. Like in Aggregate Search, the environment is cooperative. They use a Probabilistic Retrieval Model for Semi-structured Data that, by using Bayes' theorem, scores document fields based on the probability that query terms are mapped to that field. They linearly combine several type predictors (most from Federated and Aggregate Search research), alongside a new predictor that exploits document structure, where weights are gradually tuned by automatic learning methods. Used type predictions methods include *Collection Query-Likelihood (CQL)*, which aggregates all collection documents and uses this document's score as the collection score, *Query Log Query-Likelihood (QQL)*, that builds likelihood scores from previous queries to the collection, *Geometric Average*, where the collection score is the geometric mean of top n documents in collection, *ReDDE*, which scores collections on the expected amount of relevant documents, and *Clarity*, that, unlike the others, is not aimed specifically at selecting a

collection, but rather measures their performance for the query, using a language model derived from the top n collection documents, were they the chosen ones. None of the methods exploit document structure. The authors then propose their predictor, *Field-based Collection Query likelihood (FQL)*, that is similar to CQL, however, given a collection whose documents contain fields $F_1 ... F_n$, averages the probability of the query mapping to each field. Their results showed improvements, even when there was an imbalance in collection sizes.

Arguello et al. [7] focus on single vertical selection in Aggregated Search, i.e., at most one vertical chosen to retrieve documents from to complement Web search results. They investigate three sources of evidence: *Query String*, *Query Logs* and *Corpora*. Using queries by themselves, there are two approaches used: the first uses a set of rule-based triggers, i.e., a mapping between concepts that help indicate a query's intent via some key terms, while the second targets geographic features, where authors calculate the probability of a set of predefined geographic entities being in the query. Using logs from each separate vertical, they construct a unigram model to calculate the probability of each vertical being the most relevant for the query. This approach seems unfeasible in our case, as, at the moment of writing, there is no notion of verticals in the search engine and, thus, logs are not associated with anything in particular. Finally, for the last evidence source, they sample documents from each vertical and define a set of features to help describe it using existing approaches such as Clarity and ReDDE.

In a recent survey on Vertical Selection, Achsas et al. [1] state that research on this topic stalled for some years, though recently it has regained some momentum. Sources of evidence wise, the authors recognise most of the time the classification is made into two dimensions: in the first, there are three categories: *Query*, *Vertical* and *Vertical-Query* features, thus partially resembling what was identified by Arguello et al. [7] years prior. The second dimension distinguishes between pre-retrieval (e.g., click-through data, co-occurrence) and post-retrieval (e.g., hit count) features. Regarding existing work, most of what was covered appeared to try and use Machine Learning algorithms to create learning query classifiers based on previous data, which would, once again, present the limitations pointed above for usage in our case.

3.7 Summary

The emergence of new search tasks and observations from how people search motivated new research in the areas of indexing and retrieval strategies, with multiple solutions tailored for various uses cases. We expect zerozero.pt users to not differ a lot from the studies we collected in the previous chapter, hence queries should be short. Moreover, given our collection's expected content, using previous searches might help us in retrieving more relevant sets of documents. Retrieval wise, as we will describe next, our system conveys different types of documents (entities), however, most approaches in type prediction use features that we are not expecting to have (e.g., query terms that map directly to a collection and collection-specific logs). Nevertheless, some of the work developed that uses document structure is of interest, given the relational database origin of our data.

Chapter 4

Search in the Football Domain

4.1 Overview

In Chapter 1, we briefly introduced zerozero.pt, the platform that is the focus of this work, and the concerns of their owners. Their search system, which targets their internal database, wasn't producing results of the desired quality and performance. Search occurs primarily on a search bar near the top of the screen. When a user enters 3 or more characters, a top 10 of suggested best matches appears, and users can either click on one of them instantly or opt to go to the full results page (Figure 4.1 illustrates an example). Normally, one would do it if the suggestions weren't good enough, though there is the possibility of habit or fast typing: a user enters the query and instantly presses the ENTER key, not allowing the system enough time to present the suggestions that will correspond to the same results appearing in the first page of the full set. In this results page, there is another search bar with the last query made, so that further refinements can be made, if deemed necessary. This distinction is important for analysis, as some differences are expected between both: given the suggestions made, queries that didn't move past the top search bar are expected to be shorter, as users might not need to finish them fully before finding their desired result.

However, queries that weren't as popular or were more ambiguous didn't benefit from this, and were where the lack of top quality results was felt the most. As an example, Figure 4.1 shows the results for the current system when searching for [ben arfa], as well as both search bars described previously. Our underlying information need was to find the profile page of the French player Hatem Ben Arfa[1].

The first results page did not cover our information need. Indeed, the engine's first action was to take us directly to Benfica's page, wrongfully assuming from our query terms that was what we were looking for. Though a particular case, it illustrates the problem we are trying to solve. Indeed, the current retrieval algorithm works similarly to an Exponential Moving Average [92], being the result of an average between current and historic access patterns. For example, in February, visits

[1]This player's page can be found at https://www.zerozero.pt/player.php?id=9896 (accessed January 2020)

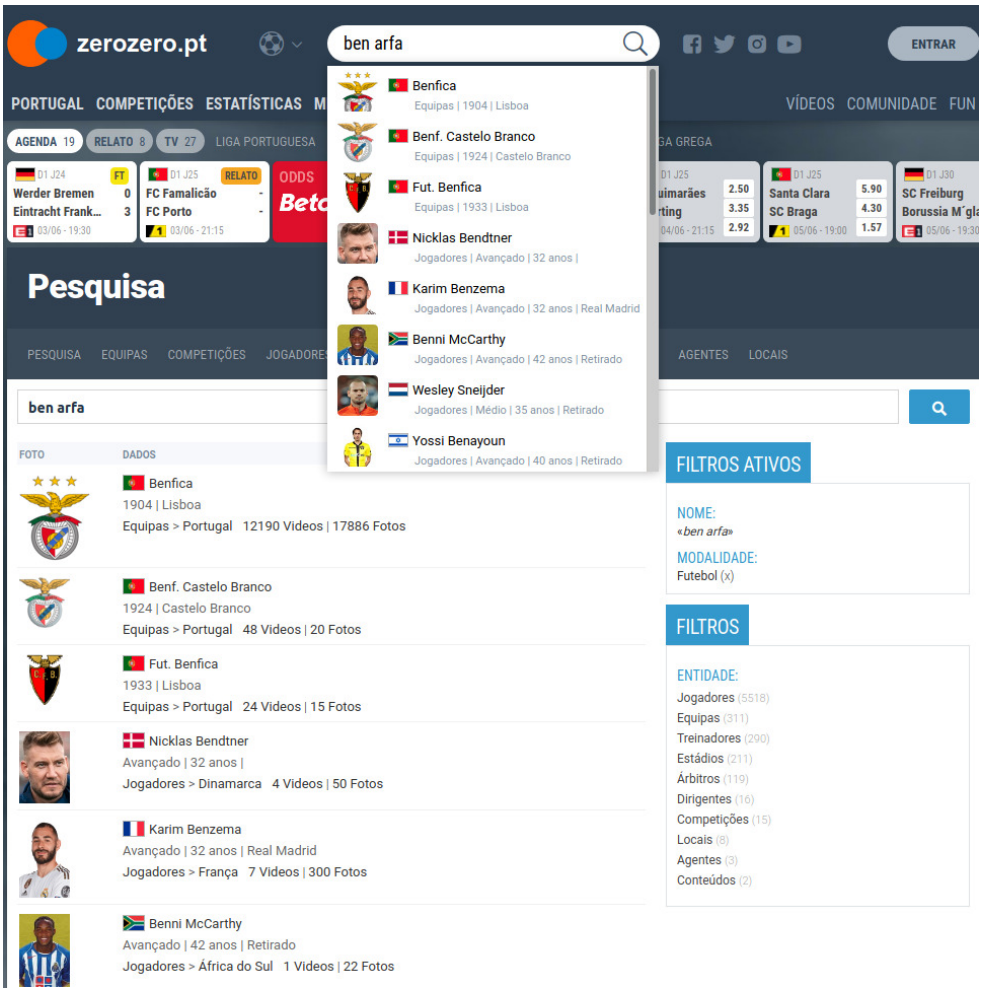


Figure 4.1: Results in current system for the query [ben arfa], as well as automatic suggestion mechanism.

on or after January account for 50% of total weight, whereas previous visits amount for the other 50%, resulting in a score *S*. In the following month, the score will then be calculated weighing 50% to visits since February and 50% to last month's historic score, i.e., *S*. This logic was then repeated for months to come. By not considering previous queries, the system might have missed out on previous access patterns that could be helpful in guiding future searchers with similar intents.

However, there were other issues associated. First, there was the need to know what kind of information we were querying and what information was available. Since we know zerozero.pt relies on a relational database, the collections will be composed of highly structured documents, with little free text, making some of the most used IR techniques hard to replicate. Second, we needed to understand what users were searching for in the platform: what types of queries were usually submitted, how did they express themselves, and were there any common or (un)expected search patterns. This analysis might help us understand specific areas where improvements need to be made, as well as establish a comparison to other studies to assess any particularities of searchers

in the football domain.

Given this analysis, there was a need for improvement in the search process itself. Unlike open web search, many of the assumptions could not be applied (e.g., the notion that a link between pages represents a vote of confidence from the linker to the linked). The project's unique context meant that enhancements weren't guaranteed to produce better results.

4.2 The Search Environment

In Chapter 3 we reviewed different search tasks that have appeared and serve distinct needs for new types of users. Each one aimed at solving a unique set of challenges. For our specific problem, we concluded that the search engine did not fully fit into one of these areas, but was instead somewhat a hybrid, containing characteristics that uniquely resembled different ones:

- *Enterprise Search* - Target collections did not grow freely as does the indexed Web for open Web search engines, but instead received new information in a controlled and limited fashion, usually manually entered by one or more system administrators. This also brought associated problems, such as a limited vocabulary;
- *Desktop Search* - Information in documents was highly structured in fields, and this structure provided information on match relevant assessment (i.e., field importance);
- *Federated Search* - There were different resource types, suggesting a separation between them, invisible to the end user, that allowed individual, independent tuning strategies (e.g., appropriate field boosts without the need for a unified schema) per resource type, without the need to worry for strategy incompatibilities between them, as independent results were normalized and merged before sending them back to the user.

These partial similarities allowed us to consider strategies from several areas. However, it also came at the cost of having to deal with additional problems, associated with the trade-offs of each area (e.g., the need to normalize document scores from different collections, as they were not directly comparable).

4.3 Proposed Solution

Our solution was two-folded, focusing on first assessing, with detail, the current situation, so that we then could propose changes in key steps of the search process that we hope lead to better results.

The first component was, in practice, the analysis of query logs. In Chapter 2 we presented the problem of Query Log Analysis and the steps involved in existing solutions. We followed these practices and, thus, would incorporate the steps proposed by Jansen [39]. One should notice that, while we were not directly responsible for the collection step, we worked closely with our partners at zerozero.pt to ensure that logs contained all the necessary information to perform the desired

analysis. In the preparation step, we would aim at eliminating queries that were either of a non-human origin or didn't meet the standards for analysis (e.g., empty or non-alphanumeric queries), as well as create a clear definition of a session. Finally, we split our analysis into perspectives, like most work collected and reviewed: term level (e.g., what terms were most used; was there a subset of terms that were reused a lot), query level (e.g., frequent queries; highest activity periods and how long did they tend to be) and session level (e.g., were sessions as short as Web search sessions; how many queries were submitted in a session). This would allow comparing our results to the ones obtained by others and assess similarities and differences of football searchers when compared to other types of users in other domains. Furthermore, we would also present a click-level analysis that is not presented as often as the other three. This analysis would allow us to conduct a more objective measurement of system quality, i.e., how well did the results serve the users' information needs. Our query logs will contain click-through information, i.e., which page(s) were clicked as a result of a particular search and what position were they in the ranked result list. Our proposal was to make use of this information and use click-through data as a relevance judgment (e.g., if a user clicks a result, we assume he found it relevant for their need; moreover, in two distinct sessions, the one where the user clicked in higher-ranked results is considered the one with a better relevance assessment by the system). In the previous chapter, we presented work from Joachims et al. [45] that exposes the dangers of using this data as absolute judgment. However, we beg to differ, given the uniqueness of scenarios like ours for this analysis: with highly structured information, users do not need to decipher through short summaries if resulting documents are potentially relevant and worthy of further exploration. The information presented in result cards is usually enough to make a final decision on a result's relevance and, therefore, the assumption that a click signals interest was expected to hold more often. Given this analysis, we would hopefully be able to detect sets of queries or patterns that we could re-subject to evaluation later to confirm our solution's quality.

In the second component, we would then focus on building a new search engine, more capable than the current one. Given the similarities to Federated Search presented previously, we would implement an independent index, using Apache Solr [70], version 8.4.1, for each target collection. However, before that, we would describe the collections at hand. The different entity types (Figure 4.1 lists some under "Filtros": players, teams, coaches, referees...) were highly structured, hence we could analyze how this structure differs, or not, between types, i.e., what fields are available to work with and how they might be helpful in the search process. This would help decide the correct schemas for each one. Given that different engines scores are not directly compatible, our solution would normalize scores before performing merging. From the similarities with Desktop Search, we would boost different document fields differently, so that matches were more relevant in some. Finally, to address the vocabulary limitations due to the Enterprise-like environment, we would complement traditional document relevance scores with previous search terms, similarly to Oakes et al. [59], though not based on such an extreme approach: while they used those terms as a substitute for document context in indexing, we'd use them as complementary information. Since documents won't have much free, unstructured text, using terms from previous queries could help other users that use them, possibly in combination with other terms, to reach these documents.

Moreover, using several sources of evidence has proved to enhance search results in previous work. Ding et al. [27] also report on the discriminative power of log indexes, helping in elevating some top results, which should meet the expectations of our users.

To assess the effect of proposed enhancements, we'd compare different versions of our engine's retrieval process to decide the better feature combination. This version would then be compared against the current production engine by using different sets of queries that reported different characteristics (e.g., popularity) collected from the QLA process conducted first. Comparison would make use of established IR metrics.

In summary, our work would aim at answering two main questions: *What are typical search patterns and behaviors for searchers in the football domain?* and *Can we improve the current system's quality by instead using a federated setup with field boosting and log term indexing?*.

Chapter 5

Search Pattern Analysis

We now present the results of our study regarding searchers in the sports domain, as a way of providing insights on the search task from a new view point, as well as providing future researchers with new benchmarks for their studies. However, before analyzing our results, we present the original dataset and describe our data preparation steps, an uncommon contribution in the reviewed studies, at least at a higher detail level. We do this for increased transparency and replicability. Finally, we analyze the processed dataset through four perspectives: the three present in most studies (i.e., term, query and session), and click-wise. The latter allows to infer, more objectively, the current system's capabilities, as well as whether there are patterns among potential improvement areas.

5.1 zerozero.pt's Dataset

The analyzed dataset concerned queries submitted to the zerozero.pt search engine for a period of a little over two weeks, between the 5th and 20th of March 2020, with tens of thousands of daily interactions on average. It was not bound by any of their sub-domains, nor by language, i.e., queries to domains other than the Portuguese[1] and from international searchers were also included. The log was stored in a compressed CSV file, with the first line corresponding to the fields header. Each entry corresponded either to an issued search, or a clicked search result: when a user went to the results page for a query, or navigated to further result pages, an entry was stored to register the search. Furthermore, each time a user clicked a result, another entry was saved, registering the current page, click position and clicked entity information. Listing 5.1 shows an artificial entry that simulates the contents of the original dataset. Each entry had 20 fields, out of which some weren't used, and others might have assumed empty values. Table 5.1 describes the main fields in the original log. In total, the original log had 400,414 entries.

[1]The project expanded, not long ago, to include international domains. For example, the Brazilian domain is https://www.ogol.com.br/.

Listing 5.1: Original log entry.

```
"id","time","date","method","page","referer","IP","sess_id","username","agent","language","bot","
    click_ranking","cache","server_ip","geo_location","search_string","tp_item","fk_item"
"1","2019−10−10 12:00:00","2019−10−10","POST","https://www.zerozero.pt/search.php?
    inputstring=my+search+here","https://www.zerozero.pt/team.php?id=12345&search=1&
    search_string=other+search&searchdb=1","255.255.255.255","aaaabbbbccccddddeeee123456","
    USERNAME","Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML
    like Gecko) Chrome/77.0.3865.90 Safari/537.36","pt","0","1","1","0","SERVER","pt","my
    search here","4","9999"
```

Table 5.1: Main fields from original log file.

Field	**Description**
time	Timestamp, in YYYY-MM-DD HH:MM:SS format, representing the moment in time when the query was made
page	URL associated with generated request
referer	URL of HTTP referer, if applicable, otherwise empty string
sess_id	Session identifier extracted from session cookie
username	Username of the authenticated user, otherwise empty string
agent	Browser User-Agent identifying the application used to submit the query
page_number	Results page viewed (0 if not applicable or result clicked from main page suggestion dropdown)
click_ranking	Clicked item position in the results page (0 if not applicable)
geo_location	Geo-location associated with the IP address that produced the query request. Stored using ISO 3166-1 alpha-2 two-letter country codes [38]
search_string	The search query, as submitted by the user to the system
tp_item	Coded type of entity clicked (e.g., 0 = no entity, 4 = player, 3 = team)
fk_item	Entity identifier (unique among its type)

Out of the non-used fields, we highlight the origin IP address. While it is commonly used to identify users, it wasn't as reliable as a session identifier and was, therefore, discarded in favor of it. The first step in our preparation was, then, to remove unwanted fields. We did this using the *cut* UNIX utility. In order to allow the terminal to correctly process each entry (since it is a text processor and, having nested delimiters used in some fields' values, would incorrectly parse entries), we used the *csvquote* [24] tool, allowing a correct processing of every field in every entry.

5.2 Preparation

Not all entries in the collected log corresponded to queries of human origin or of interest to us. Thus, there was a need to filter out machine-originated interaction. Existing literature pointed us to two main steps: removal of queries whose User-Agent string matched a non-human origin [21] and sessions that had an unusually high number of queries and, therefore, could be assumed to fall under the same category [69].

5.2.1 Unwanted Queries Removal

Queries of non-human origin appear for different reasons in logs. They can be the result of a Web crawler's actions that was updating their index entries for an open Web search engine, or malicious spyware with worse intentions (e.g., injection flaws, still considered the biggest threat in Web applications [60]). Since we were studying human search behavior, these queries mustn't be included in our analysis. However, bot removal is not a solved problem. It is not possible to affirm that, given a list of bot candidates, all remaining entries correspond to human searchers. The converse is easier to assure: given the same list, we may be able to say that all entries identified were of non-human origin. The easiest technique to identify non-human queries given single entries from the log file is the analysis of the User-Agent string. Indeed, when web spiders or crawlers perform requests, often they leave identifiable traces in this field.

The matching was done via regular expressions. We used the *awk* UNIX command, since we could provide a separator and work with the field directly. For this matching process to be effective, we needed to match two conditions: first, the bot list should be comprehensive; on the other hand, it mustn't be too restrictive, as we wanted our expressions to have high discriminative power. In other words, if some terms helped identify a family of User-Agent strings, and did not conflict with normal User-Agents, we would use those instead of longer, more specific strings that contained Operating System or version information. Some terms were generic enough that we could even decouple them from the type of bot. Take the terms *spider* and *bot*, for example. Both discriminate against several different bots, and nothing else. To build this list, we used predefined User-Agent collections with known bot expressions [77, 84]. Moreover, we refined this list by manually identifying and inspecting remaining suspicious User-Agent strings. The full list used is shown in Appendix A. In the end, 9,215 queries were removed as a part of this process. This represented only about 2.3% of all queries.

Despite filtering through the User-Agent field, some undesired entries still remained. More specifically, empty queries were of little interest for analysis. Moreover, there could have been bot queries that weren't removed as a result of the previous process, and there could be patterns that helped to identify them. For the former, we filtered out all entries whose search text field didn't contain any text for analysis. This amounted to 0.57% of the dataset. For the latter, upon inspection of the intermediate log file, there was a presence of entries whose search queries used characters not representable using ASCII. Their content, after translation, was out of context, given the search engine's goal. Some examples of translated queries included [searchhealth], [redA

wake-up call for all humanity! Scientists have been in the sea] and [Application for Business negotiations | Yanase Regular dealer of foreign and imported vehicles]. It was then decided to consider all these types of queries as irrelevant. With this step, 40 extra entries were removed.

5.2.2 Session Definition

The next step involved defining sessions in our dataset. While we had a field with a session cookie identifier, using it by itself might have not been enough to isolate user sessions, as there was the possibility of a user trying to satisfy different user needs using the same browsing session. Therefore, we instead relied on a temporal cut-off between queries to maximize our sessions' duration. We used a 30 minute window that has been adopted before [21, 80, 82]. The session definition algorithm worked as follows: first, the dataset was sorted on the session identifier field and, in case of a tie, on timestamp. Then, given that the first query had a session key of 1, every other entry was processed sequentially, such that, if the session identifier differed, or both entries were separated in time by a period of at least 30 minutes, we increased the session key by 1, otherwise it stayed the same. The resulting session key value was associated to the current entry, before moving to the next one. This outputted an extra field in each entry with this new session key.

With sessions identified and isolated, we proceeded to eliminate longer sessions that were a sign of non-human interaction with the search engine. Like Silverstein et al. [69], we used a cut-off of 101 queries, that is, sessions with 101 or more queries were discarded. While we can't guarantee that we wouldn't discard some human sessions, 101 queries within a single session seemed unlikely, even for a curious and persistent searcher. In the end, we were able to filter out 1.6% of all entries distributed across 30 sessions. We concluded that bot interference is an issue felt with less impact in this search engine, a conclusion that we didn't anticipate, however, should not come as a surprise: our search engine does not target the whole web and, therefore, wasn't likely subject to as much exposure. Our final dataset for analysis contained 382,323 entries, representing approximately 96% of the original's size.

5.2.3 Query Standardization

Finally, all queries were preprocessed so that subtle differences that introduced no semantic distinctions were removed. This allowed to more correctly group queries that referred to the same thing, but might have been sent to the system in different ways. Three operations were then chosen to accomplish this. First, all leading and trailing whitespaces were removed. Furthermore, all multiple whitespaces were compressed into a single delimiter (e.g., [this query] is equivalent to [this query]). Second, all queries were converted to lowercase, as casing does not have any special meaning in this context. Last, all characters, when applicable, were converted into their ASCII representation, using the UNIX *iconv* tool. Users are not always consistent in their writing: some may write out terms with accents, others leave them out, as often systems already recognise

both versions as the same anyway. This way, we facilitated our analysis process, as well as assured consistency and correctness.

5.3 Analysis

Given a dataset ready for assessment, the analysis is now to be presented. It was tackled according to four main levels, three of which identified during our background and related work review: term, query and session. In some cases, there were further sub-partitions. For example, in the current production engine, users were suggested some top results when writing a query and could opt to click on one immediately instead of going to the full results page. Over 57% of log entries corresponded to these interactions that occurred on the main page and not on the full results page. We then present, both at term and query level, measurements not only for the full log, but also of this log subset, which we refer to as *Main Page (MP) Subset*.

The extra level corresponded to a click analysis, something not found on most previous work, due to this information not being easily collectable. This helped us take a perspective not often explored, as well as have a more objective basis for evaluating the system's quality. We will present our results, compare them to other studies when relevant, and otherwise explain our new additions and perspectives.

Since our dataset contained more than query entries (e.g., clicks and pagination), and to avoid wrongful bias in the results, in some metrics we only considered unique query entries. In other words, entries that corresponded to a user navigating through result pages or clicking a result were discarded. For this, when necessary, we only kept entries that either had a *page_number* of 0 (meaning they didn't go past the main page search) or that had a combination of *page_number* of 1 and *click_ranking* of 0, meaning entries registered when visiting the first full results page. At term and query level, this filter was always applied. At session level, this consideration was metric dependent. Unless stated otherwise, the full dataset was used. Finally, at click level, the source problem was different, as we were interested only in click entries and, therefore, applied a filter to only consider rows with a positive *click_ranking* value.

The whole process was automated as much as possible, so that, when new input was available, there was only the need to modify its source. Appendix C describes the QLA pipeline adopted in this study.

Table 5.2 presents a starting overview of some of the main metrics measured at each level of analysis, before we more thoroughly go into each one. Some patterns seen previously can already be observed, such as short queries and sessions.

5.3.1 Term Level

We have previously defined a term as a whitespace-bounded sequence of characters that is part of a query. For the purpose of this analysis, we were more rigorous and did not consider all types of characters in this sequence. The main reason was the presence of possible attempts at advanced operator usage. Despite the system currently not supporting any type of advanced search via the

Table 5.2: QLA main statistics summary.

Analysis level	Metric	Value
General	Nr. Queries	324,157
	Nr. Terms	435,641
	Nr. Sessions	190,980
	Nr. Clicks	270,496
Term	Mean Characters per Term	6.44
	Unique Term %	10.72%
	Never Repeated Terms %	5.53%
Query	Mean Terms per Query	1.34
	Unique Query %	21.63%
	Never Repeated Queries %	13.13%
Session	Mean Queries per Session	1.87
	Mean Session Duration	3min 18s
	Mean Nr. Pages Visited	1.04
Click	Mean Clicked Ranking	1.75

query text, users still tried to use common operators learned elsewhere. We looked into this in more detail at query level. For now, after isolating terms by whitespace boundaries, we removed operators in two ways: first, deleted all occurrences of explicit concatenation (e.g., [queries+like+this]), replacing them with a whitespace. After this, we deleted any possible remaining boolean operator characters, such as other plus signs (used to force term presence), as well as hyphens (used for negation) and quotes (used for phrase querying).

Finally, we built a list of stopwords, so that a parallel could be drawn between a full term dictionary and one without these terms that may present little semantic meaning, just for comparison. The stopword list was based on the main Portuguese prepositions and can be found, in full, in Appendix B. Table 5.3 presents an overview of common statistics obtained for analysis at term level.

Out of the three most common levels, term level was the one less present in previous studies, hence some of the metrics offered little benchmark comparison. The first noticeable difference was in the total number of terms used. When isolating the Main Page Subset, the value dropped by over 40% when compared to the full dataset. This tendency was kept when discarding stopwords from the term dictionary. One reason for this discrepancy is the higher likelihood need of people who visit the full results page to rewrite their query.

When calculating unique term percentage, a value of 10.72% was achieved. This percentage

Table 5.3: Main term level statistics summary.

Metric	Dataset	MP Subset
Number of terms	435,641	260,949
Number of terms (no stopwords)	424,616	258,395
Unique term percentage	10.72%	13.20%
Never repeated terms percentage	5.53%	6.81%
Mean characters per term	6.44	6.16

rarely agreed between studies, and ours was no exception. The closest match was Yi et al. [99], who reported 9% and 14% for history and psychology databases, respectively. Other studies achieved both lower (4.5% from Chau et al. [21]) and higher (26% from Wang et al. [86] and Tang et al. [78]) values, hinting that this metric was very context-dependent, especially in domain-specific search engines. When analyzing the percentage of terms that made up the long tail, i.e., only showed up once, values were reasonably consistent (a little over 1 percentage point increase in the MP Subset). When only considering unique term occurrences, the value rose to just over 51%. In other words, over half of the vocabulary used had never been seen before. More common spelling errors, as well as uniquer entity names are possible reasons for the difference.

Regarding the number of characters used per term, an overall value of 6.44 was achieved, dropping to 6.16 in the MP Subset. Once again, but from a different metric, we saw how searches tended to be shorter on those circumstances. The values remained equal, to the extent of presented precision, even when discarding stopwords. This is consistent with the difference shown in the number of terms: stopwords represented only around 2% of total terms. Furthermore, in this context, searches were likely to contain entity names that seldom have these terms present. Some studies argue that this metric is dependent on the language of the search interface [64]. In our case, there was no clear and distinct language, as entity names remain mostly the same throughout languages. Exceptions included team names (e.g., if they have city names in them), where we expected our data to be biased towards Portuguese vocabulary. Figure 5.1 shows how term size, in characters, was distributed.

As stopwords are usually of lower length (in our stopword list, the maximum size was 5 characters), the distributions between full term dictionary and no stopwords only differed in the lower end. Results showed signs of a normal distribution, although further tests would be necessary to confirm this hypothesis. Indeed, in that kind of distribution, it would be expected that, given an average μ and standard deviation σ, the range $[\mu - 3 * \sigma, \mu + 3 * \sigma]$ contained 99.7% of all values. This distribution had a standard deviation of 2.39 characters. Hence, the corresponding range contained 99.73% of all values, a value very close to what would be expected. The maximum size registered for a single term was 103 characters, corresponding to a never repeating term, much like most terms with 20+ characters.

Another distribution pointed out while reviewing previous work concerned term cumulative frequency. With it, we could assess the variety of vocabulary from another perspective, by check-

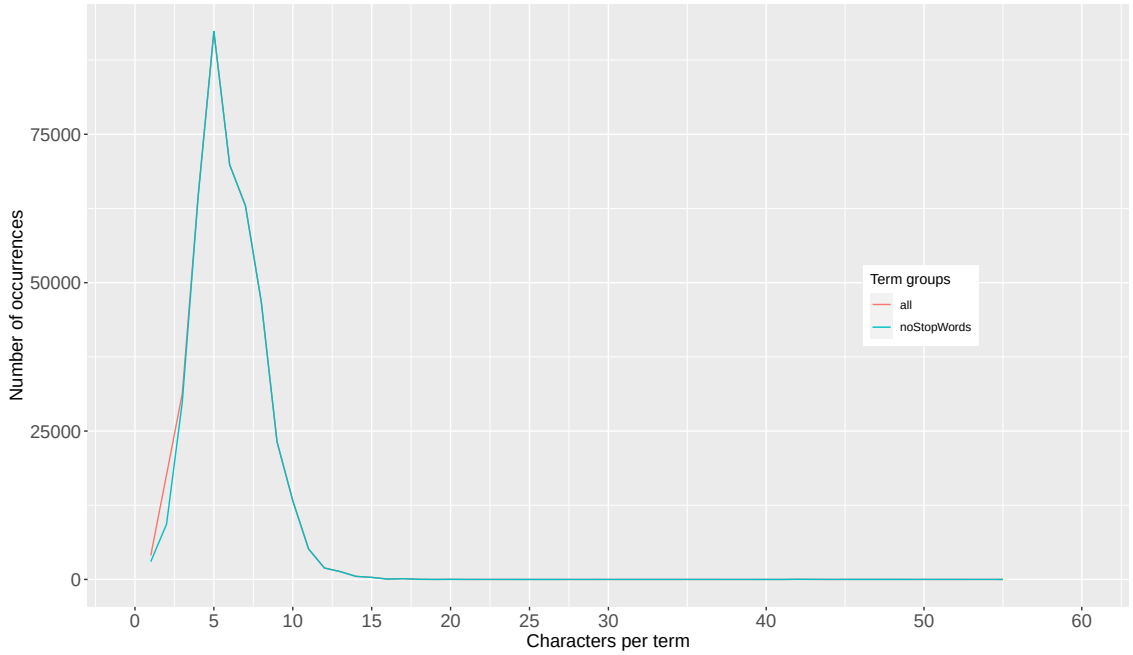


Figure 5.1: Terms per character distribution.

ing how much influence the top terms have on the overall frequency. This could also help in designing caching mechanisms for more efficient answering [2]. Figure 5.2 shows this cumulative frequency distribution. To calculate it, terms were sorted in decreasing order by their frequency.

The distribution appeared to fit a power law, differently than the inversely proportional relation obtained by Wang et al. [86], with no noticeable differences when removing stopwords. As for caching, storing the top 2% of terms would answer 53.6% of all queries with those terms. When using the MP Subset, the distribution followed a similar pattern, but with a decrease in caching possibilities: a coverage of only 45% under the same conditions. We also calculated the coverage percentage for the top 100 terms, as done by Spink et al. [74]. They arrived at a value of 19.3%. Here, the value was 22.3% (19.5% in the MP Subset). Given that their study targeted an open Web search engine, the range of information needs was expected to be higher. Therefore, a bigger vocabulary was likely needed and a lower coverage percentage followed.

Last, we looked at the most frequent terms themselves to understand what users used more frequently when searching, by collecting the top 20 most used terms. Moreover, since many entities were usually addressable by two terms, and in an attempt to predict common queries, we also collected the top 20 query bigrams. A query bigram is a sequence of two consecutive terms found in a query. Tables 5.4 and 5.5 illustrate both these top results.

A stopword was the most used term in this log. Naturally, it was the most used stopword, and the other two that appeared on the list were contractions based on the first one (e.g., *do* comes from the contraction of preposition *de* with article *o*). Together, they accounted for two-thirds of all stopword occurrences. Some terms could form queries by themselves (e.g., *porto*, *benfica*), while

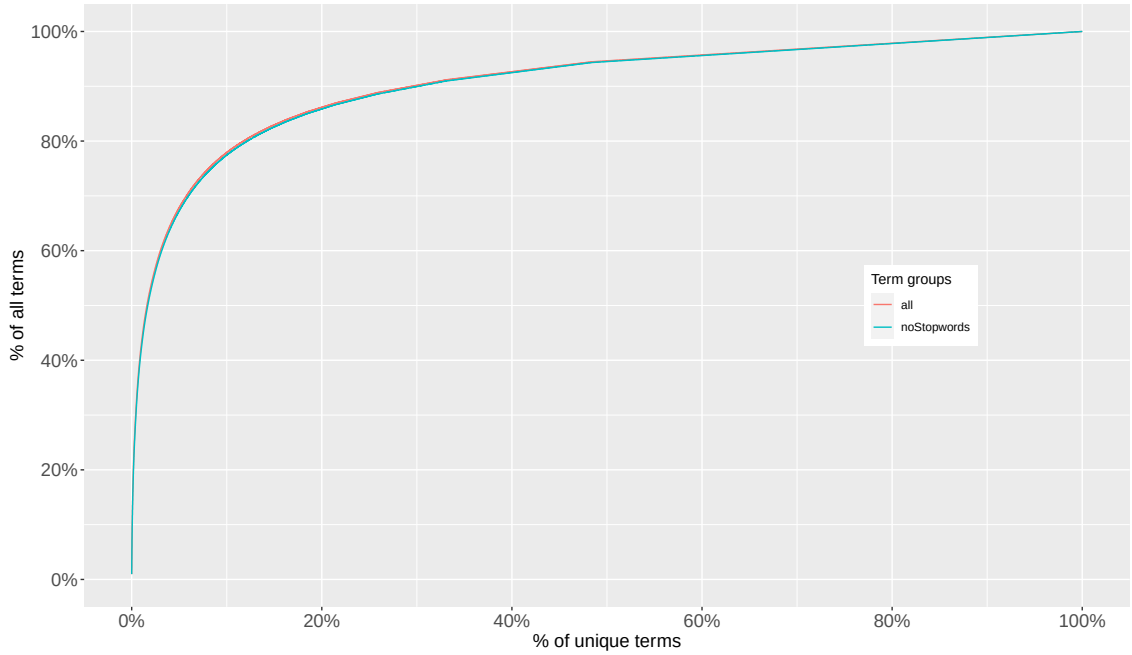


Figure 5.2: Term cumulative frequency distribution.

others should be most used to form expressions (e.g., *liga*, *campeonato*). This was confirmed by the top 25 bigrams list, where these isolated terms disappeared, and others appeared together to form entity names. With the bigrams list, we got a better understanding of what users searched. Some entries were expected to be there regardless of the period analyzed, but others were likely a result of a spike due to the collection period.

Some bigrams appeared to be part of a larger expression (e.g., *campeonato de*), however, not even the top bigram was half as frequent as the lowest top term. Therefore, pursuing trigrams would likely lead us to insignificant frequencies that would make comparison difficult.

5.3.2 Query Level

Analysis at query level is usually more thorough than at other levels, as well as being more popular. Often the focus of most studies regarding Query Log Analysis, due to the equivalence of one entry corresponding to one query, it is also where most high-level search distributions are presented. Contrary to term level analysis, there was no further processing, other than those mentioned during the preparation phase. Table 5.6 presents some general statistics measured at query level.

Given the time period over which the dataset was collected, the total number of queries surpassed other domain-specific search engine studies. The ones with higher entries were the Univ. of Tennessee study by Wang et al. [86], with over 500 thousand records, but over a span of 4 years, and the work of Chau et al. [21], that spanned over half a year. Only around one-fifth of queries were unique, symbolizing some similarity between users on expressing information needs. This

Table 5.4: Top 20 most frequent terms.

Term	Frequency	Percentage
de	4,222	0.93%
liga	4,101	0.90%
porto	2,273	0.51%
benfica	2,203	0.49%
santos	2,093	0.48%
bruno	1,842	0.45%
braga	1,841	0.42%
joao	1,810	0.41%
flamengo	1,762	0.41%
sao	1,745	0.40%
real	1,734	0.39%
pedro	1,662	0.38%
sporting	1,635	0.38%
da	1,572	0.36%
paulo	1,532	0.36%
do	1,524	0.36%
silva	1,502	0.35%
rio	1,494	0.34%
atletico	1,382	0.33%
campeonato	1,329	0.31%

Table 5.5: Top 20 most frequent query bigrams.

Bigram	Frequency	Percentage
rio ave	606	0.50%
sao paulo	525	0.43%
ruben amorim	518	0.43%
liga dos	385	0.32%
de portugal	359	0.29%
real madrid	357	0.29%
jorge jesus	338	0.28%
campeonato de	327	0.27%
da silva	324	0.27%
copa do	318	0.26%
bruno fernandes	299	0.25%
fc porto	278	0.23%
af porto	232	0.19%
liga nos	206	0.17%
sub 19	202	0.17%
santa clara	195	0.16%
manchester united	192	0.16%
porto salvo	186	0.15%
gil vicente	182	0.15%
vila real	178	0.15%

result was consistent with other studies (both Fang et al. [31] and Ribeiro [64] reported similar values, and Yi et al. [99] came up at 25%). Over 13% of queries appeared only once throughout the collection period, a value that dropped close to 12% when considering only main page searches. To complement this, and similarly to what we did at term level, we analyzed the cumulative query frequency distribution as a way to assess how representative the top queries were when compared to the whole dataset. Figure 5.3 shows this distribution.

The frequency distribution followed, once again, a power law. There were no noticeable differences between a dataset with and without filtering stopwords: with their removal, there was but a very small increase in coverage. By once again assessing potential caching benefits, we could verify that, by storing the 0.4% most frequent queries, we could cover 24.58% of all queries. Moreover, if we extended this caching to the top 20%, the value would rise to 77.68%. The study by Ribeiro [64], with similar analyzes, obtained around 50% and 80% for these two thresholds, respectively. We concluded, then, that the top queries weren't as representative: even though, at 20%, values were similar, this was expected behavior as the threshold rises. However, this was somewhat expected, given the open Web nature of that study. Silverstein et al. [69] viewed frequency distribution from another perspective and calculated the representativeness of their top 10

Table 5.6: General query level statistics summary.

Metric	Dataset	MP Subset
Number of queries	324,157	221,742
Unique query percentage	21.63%	19.93%
Never repeated queries percentage	13.13%	11.82%
Mean characters per query	8.21	6.98
Mean terms per query	1.34	1.17
Mean terms per unique query	1.69	1.38

queries, arriving at a value of close to 1%. For us, the same statistic produced 3.69%. Though over 3 times higher, one must consider that Silverstein et al. had a much larger dataset. It would be interesting, in a future analysis, with more data, to assess the evolution of these values, in order to check if there is some convergence towards similar results.

These results raised other questions, namely about exactly how frequent were new searches, i.e., searches never seen before. Knowing that it was impossible to obtain a reliable value without full access to the search engine's full history, we could still provide an estimation by assessing how frequent new queries were over the collected period. We then analyzed the new search ratio on a daily basis. In other words, for a given day, we calculated the ratio between the number of queries that hadn't been submitted in any prior day (taking into account their frequency on that day) and that day's total number of queries. Figure 5.4 shows the results obtained.

Results showed a clear stabilization of this ratio as we moved forward in time. Considering the first day as our beginning of time, it only took 3 days for the ratio to reach a value of 25.29%, which converged to around 22% over the next days. From this point on, the difference was never as big (never above 2 percentage points), with a maximum of 23.56% on March 18th. For comparison, Google reported an average of 15% of new searches every day in 2017 [35].

Regarding query length, we first looked at the number of characters used. On average, queries had 8.21 characters, a value that dropped to just under 7 characters for the MP Subset. It was interesting to see that these values did not differ much from the mean number of characters per term seen previously (6.44 and 6.16, respectively). This already showed a tendency for shorter queries. The difference from term level values could be attributed to several reasons. For one, naturally, queries are longer than terms, despite the tendency for query shortness. Moreover, in queries, there are characters that do not appear at term level, namely word boundaries (e.g., whitespaces) and operators, such as double quotes.

By looking at query length by number of terms, we could affirm that queries were, in fact, short. Aside from the main result of just 1.34 terms on average, we saw the value rise to 1.69 when considering the set of unique queries, meaning there was influence of more frequent, shorter queries on the general value. Furthermore, it was a shorter value than most reference studies. AltaVista [69] and Excite [74] studies reported 2.35 and 2.4 terms per query, respectively, while AOL even saw a rise from 2.2 [10] to 2.7 [11] terms. Even when considering domain-specific

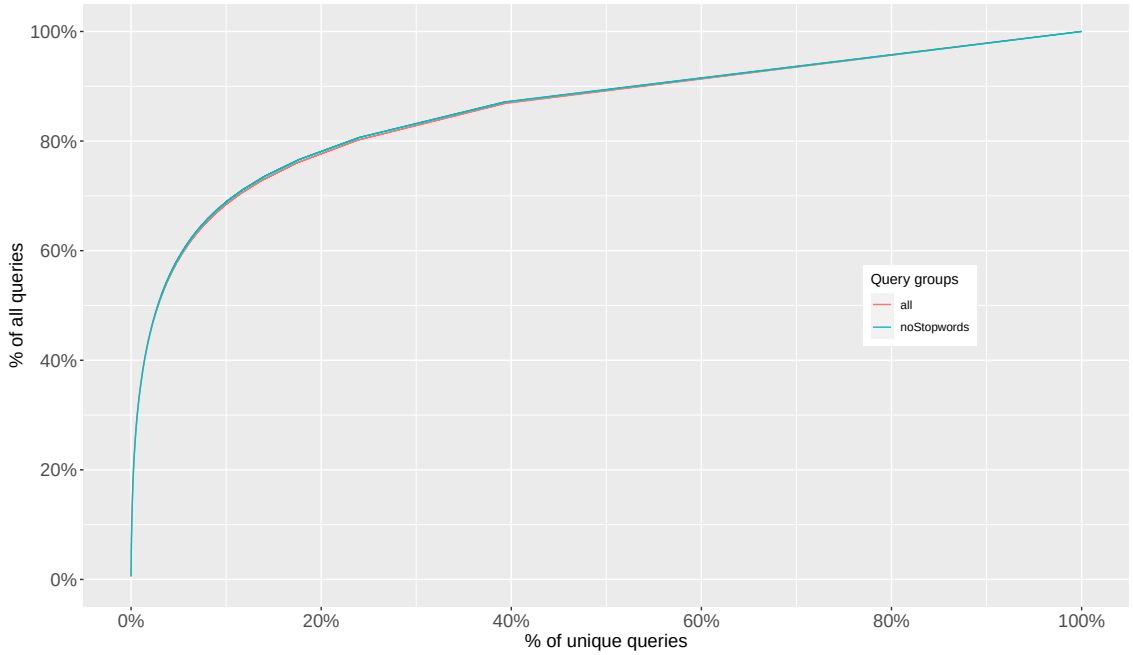


Figure 5.3: Query cumulative frequency distribution.

search engines, Yi et al. [99] reported over 3 terms for both history and psychology databases. Only in the INDURE [31] study did this value also go below 2 terms per query, though it wasn't far off (1.96). Table 5.7 shows the full distribution of the number of terms per query.

Table 5.7: Terms per query distribution.

Terms	**Frequency**	**Percentage**
1	240,028	74.05%
2	67,198	20.73%
3	11,150	3.44%
4	3,541	1.09%
5	1,283	0.40%
6	457	0.14%
7	258	0.08%
8	100	0.03%
9	45	0.01%
10+	97	0.03%

Results showed that nearly three-quarters of queries had only one term. It then seemed that users tended to search for entities only by their common name(s), not often going to greater lengths to provide more verbose queries. For popular entities, one term might have been enough: teams often have famous one term names or abbreviations/nicknames, and the same goes for players.

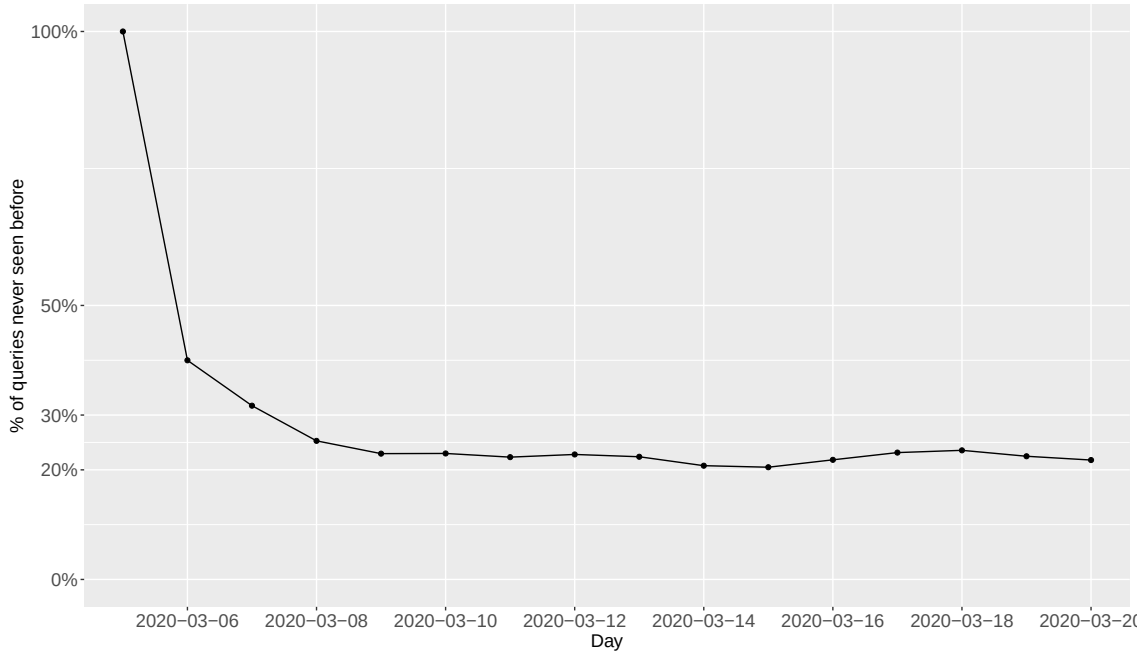


Figure 5.4: New searches ratio over a daily basis.

However, when there is overlap (e.g., in player names) and entities aren't as popular, this line of thought might not always produce the desired results.

It was also of interest to analyze how users reformulated queries throughout a searching session. We identified four main types of query: *Initial* for the first query detected in a session, *New* for a query not seen before in the session, *Identical* for a query that had been submitted previously, and *Modified* for queries that matched some terms of a previous one, but not fully. Table 5.8 shows an overview of obtained results.

Table 5.8: Query types distribution.

Type	**Frequency**	**Percentage**
Initial	173,468	53.51%
Identical	120,914	37.30%
New	22,191	6.85%
Modified	7,584	2.34%

Initial queries assumed a majority share of the query type distribution. This showed that users usually did not deem fit or necessary the effort to reformulate a query to tune the result list to their needs. The value was higher than the one obtained by Ribeiro [64], who achieved 34.72%. As per *Identical* queries, the values matched more closely (33.32% in their study), a value nonetheless high (Chau et al. [21] obtained a 12.4% ratio of repeated queries). With a much lower share, *New* and *Modified* queries appeared with 6.85% and 2.34%, respectively. Silverstein et al. [69]

saw that, in the AltaVista engine, close to 35% of queries within a session were completely new. Interestingly, when distinguishing authenticated users, *New* queries were the ones that had a bigger ratio compared to their respective total (21.25%, with all others equal or below 15%). Considering authenticated users are likely long time users, we saw that these searchers may have learned that, if a query didn't work at first, it was better to discard it and try a new one.

After *New*, *Modified* queries came in second place when distinguishing authenticated users, and they provided another way of looking at how users learned to reformulate. In other studies, this type was more frequent (e.g., Silverstein et al. [69] reported 10.2% of cases involving term deletion or addition). Table 5.9 shows the distribution of the difference in number of terms in this type of queries.

Table 5.9: Difference in number of terms for *Modified* queries.

#Terms difference	**Frequency**	**Percentage**
-5 or less	13	0.17%
-4	22	0.29%
-3	76	1.00%
-2	284	3.74%
-1	989	13.04%
0	4,269	56.30%
1	1,380	18.20%
2	396	5.22%
3	99	1.31%
4	36	0.47%
5 or more	20	0.26%

Results showed that users tended to keep query length constant, with over half performing term substitution, i.e., changing some, but keeping the overall total the same. Moreover, specialization by term addition was preferred over generalization by term deletion. This was expected, as users generally start with broader queries, which require less effort and may sometimes achieve similar enough results. When isolating authenticated entries, the distribution followed a similar pattern, with 81.04% of entries corresponding to a difference of at most one term. These results were consistent with findings by Ribeiro [64] and Wang et al. [86], who concluded that users tended to modify queries one term at a time. The latter identified this process as "mental models of Web searching and learning". Indeed, this appears to be a pattern not specific to any domain, but present everywhere.

Advanced operator usage was also a frequent analysis made on QLA studies. Advanced operators include not only explicit boolean operations (e.g., AND, OR), but also phrase queries (i.e., encapsulating an expression in double quotes) and wildcards (e.g., query [wh*] matches *what*, *why* and *where*), for example. As of the time of writing, the engine does not support any advanced operators. Still, it was of interest to assess usage attempts in order to understand what searching

patterns users acquired from other engines and assumed as present by default. Results found that attempts were not common, with only 0.3% of queries attempting to use at least one of these operators. The most common was phrase querying, with around 70% of the overall share, followed by the explicit conjunction operator (e.g., [queries+like+this +or +this]), with a ratio of just 4%. When considering only authenticated entries, it is the latter that has a bigger relative percentage to their total, with 97% of entries coming from authenticated users. This might represent an attempt at users who tried to force the system into acknowledging terms that they felt weren't being deemed relevant. Overall, there was still a pattern similar to other studies whose engines had advanced operator support: 1.5% for other Portuguese studies [64], while Chau et al. [21] arrived at 3.4%, demonstrating the little reach these operators have, the lack of value given to the effort trade-off, or even lack of necessity in order to obtain satisfactory results by users. In both cases, the most common operator was the phrase query.

It was also interesting to note the presence of URLs in queries submitted, more specifically URLs associated with the engine's website. Given these URLs' structure, it was possible to extract the entity associated with it through simple regular expression matching. We found that, while this presence was limited overall, having only 25 occurrences, 21 of these corresponded to URLs concerning news pages. News aren't searchable entities in the analyzed engine. zerozero.pt owners have explained that this hadn't always been the case and, when present, news would overflow the results list, due to their overwhelming presence. Moreover, we hypothesized that their unstructured nature and abundant text, fitting better into the IR paradigm, would also help explain the phenomenon. As such, there is now a dedicated news search engine[2]. Results could be interpreted in many ways. While some users might have been unaware of this feature, erroneous copy and paste might just as easily explain the presence of these URLs. There was only one other entity mentioned: player, with 4 occurrences.

At a higher abstraction level, it was possible to assess distributions regarding information associated with the query, aside from the text itself. Despite primarily being a Portuguese website, as we stated before, the project has expanded to several foreign domains and collects visits from several countries. As our log contained geolocation information, we had access to the request's country of origin. While not infallible (e.g., the searcher could be using a VPN - Virtual Private Network), we didn't expect these edge cases to influence the overall results too much. Table 5.10 shows the top 10 values found in the geolocation field, translated from their two-letter code [38].

Portuguese searchers were, unsurprisingly, a majority, with a close to 60% share. With nearly half of that value (30%), Brazilian searchers came in second place, with Spain and France after, though with much lower shares. All of these countries hold close ties to Portugal, not only due to their history and geographical proximity, but also the tendency of players moving between the respective countries' main leagues. Another trend in the last few football seasons is the arrival of Mexican players to top Portuguese clubs, which may explain the presence of the Central American country. As for the other non-European entry, the USA, we speculate that their proximity to Mexico was the main reason for their high frequency. Entries with an empty value for the geolocation

[2] https://www.zerozero.pt/news_list.php (accessed April 2020)

Table 5.10: Top 10 values for geolocation field.

Country	**Frequency**	**Percentage**
Portugal	192,116	59.27%
Brazil	91,106	28.11%
Spain	4,099	1.26%
France	4,030	1.24%
USA	2,750	0.85%
Great Britain	2,726	0.84%
Switzerland	1,927	0.59%
Mexico	1,725	0.53%
Angola	1,707	0.53%
Germany	1,310	0.40%

field were not considered. They accounted for 1.35% of all queries.

Given the timestamp information on each entry, it was also possible to analyze query distribution on several levels of granularity. We started by looking at a daily distribution for all days in the collected period. Figure 5.5 shows this distribution.

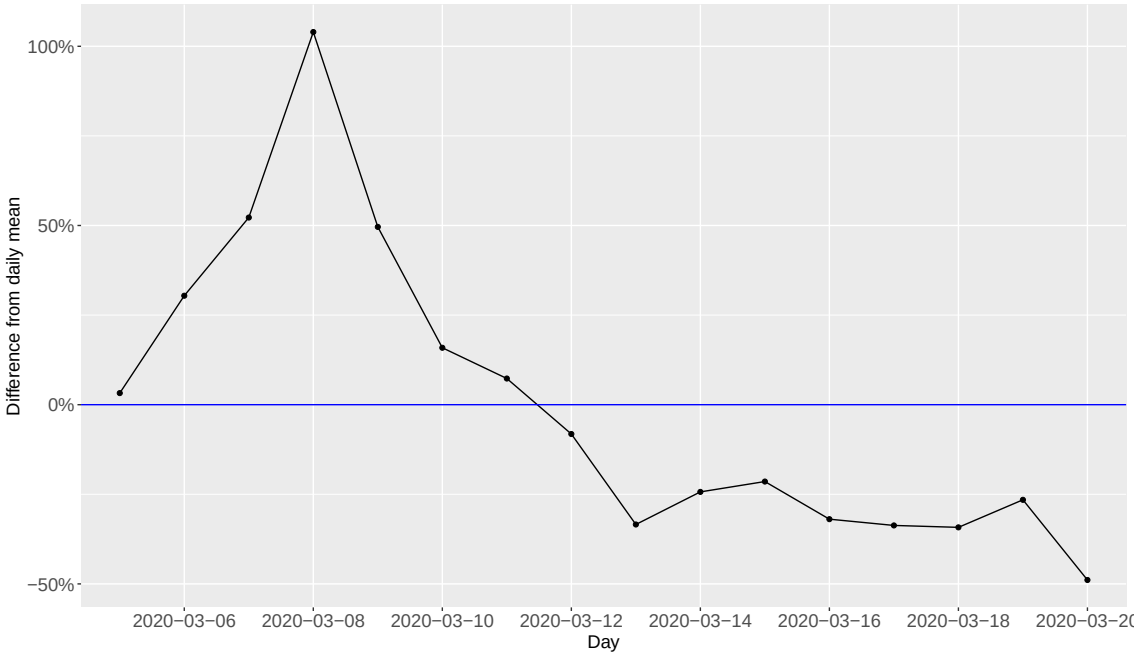


Figure 5.5: Daily query distribution (relative difference to mean).

On average, 20,260 queries were submitted per day. However, this average came as a result of two very different weeks. On the first one, activity corresponded to what would be expected of a typical week: nearer average queries, with localized spikes on weekends, as most matches

take place during that period. This explained the bigger query frequency on March 7th and 8th. Moreover, there were decisive matches concerning the Portuguese league during those two days. With a frequency of just over half as much as the daily mean, March 20th represented the least active day overall. However, during the collection process, it was not possible to capture the full 24 hour period for both March 5th and 20th. Therefore, their total frequency was likely inferior to reality.

Starting on March 12th, there was a significant decline in query frequency, with almost all days registering a near 25% drop from the mean. This was caused by the 2020 COVID-19 pandemic outbreak that first reached Portugal in the beginning of March [95]. Given its spread in little time, on March 12th, the entity that regulates Portuguese football suspended all competitions until further notice [26]. With no football, most searchers lost interest or motivation to perform searches. We later analyze the impact of the pandemic in the search behavior of the remaining users.

By abstracting into a weekly basis, we obtained the pattern shown in Figure 5.6. With no surprise, the period surrounding the weekend was the one with bigger activity, corresponding to the preferred day for matches to take place. Sunday came in first with over 20% above average, with Thursday in second place. This last result was somewhat surprising at first. However, Thursday was the only weekday appearing three times during the collection period, with all others appearing only twice, hence the higher frequency with another drop following it. We would expect a typical pattern to see a slow rise throughout the week. This was the kind of pattern obtained by Beitzel et al. [10] in their AOL search log analysis: a peak on Sunday, following by a gradual decrease throughout the week, until a minimum was reached on Fridays. Interestingly, other Portuguese studies [64] obtained opposite results: weekends corresponded to lesser activity periods, and peaks were found at the beginning of the week (i.e., Monday). We could interpret this in many ways. A possibility was a context shift occurring on weekends: sports, and in particular football, is a popular sport amongst the Portuguese and, when it is on, it may take most of their attention.

It was also relevant to assess distribution on an hourly basis. However, here, it was important to separate between different geographical locations, due to their timezone differences. The log stored all timestamps using the Portuguese timezone at the time of the request, possibly creating unrealistic information for queries from other countries. While the same could be said for daily and weekly distributions, as time differences on late night hours can transcend days, query frequency at those transitional hours was much lower and, therefore, possible deviations likely haven't affected the obtained results significantly. We plotted the hourly distribution for the top 4 geographical locations identified previously (see Table 5.10). Given that some countries were large and had different timezones for different cities, and that we did not have city information, we approximated the country's timezone by setting it to the respective capital's (e.g., for Spanish queries, we added 1 hour, as Madrid is one hour ahead of Lisbon). Figure 5.7 shows the results obtained.

Some countries had more distinct patterns than others. Portuguese and Brazilian searchers had similar behavior until early afternoon (2 PM), with the former peaking around 5 PM, while the latter achieved theirs at dinner time. French searchers had more spikes than other searchers. Some

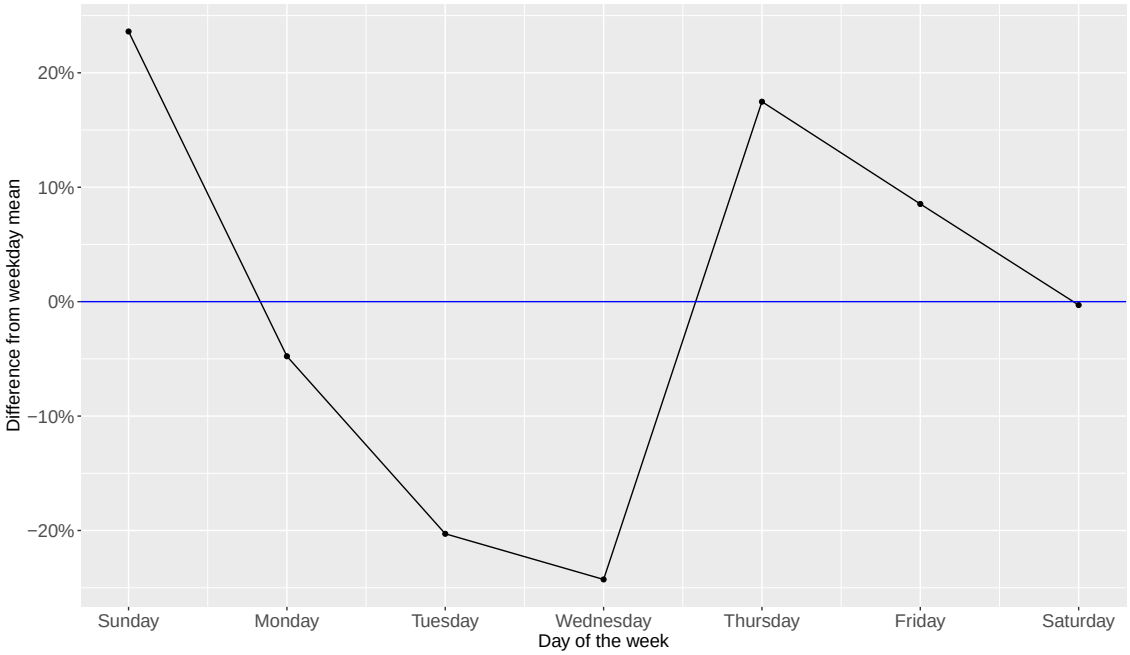


Figure 5.6: Weekly query distribution (relative difference to mean).

(at 1 PM and 8 PM) likely corresponded to meal breaks. The other, between 4-5 PM, was near the end of a typical day's worth of work. Finally, the Spanish were the most distinct. The earliest risers and the least active at night, they peaked considerably before lunch time, at 11 AM. The drop that followed could be interpreted as a sign of the common post-lunch nap (locally known as *siesta*). Hourly distribution per country was not common in other studies. Our best approximation for a comparison would be to consider only Portuguese searches, the majority of our dataset, in which case results were consistent with other findings [64].

Last, we looked at the top queries submitted to assess what were most frequently searched by users and how it compared to the top terms and query bigrams previously presented in Tables 5.4 and 5.5. Table 5.11 illustrates the top 25 queries.

Results showed some similarities between top queries and top terms. This was expected due to queries being typically short, with a term by itself being a query most of the time. Bigrams weren't present as much, however the top bigram identified ([rio ave]) was also present, even if in the last position. The top 5 queries included big Portuguese clubs and also Brazilians Flamengo, who have ties with Portuguese manager Jorge Jesus. In fact, it was interesting to see some tendency for South American-related entity searches, with nearly half of entries corresponding to them. This was likely a non-recurring spike due to the COVID-19 outbreak which we will look into with more detail next. Entity type wise, most queries appeared to target teams, with only three entries concerning other types ([liga] for competitions, [ronaldo] and [pedrinho] for players). Finally, we saw that the top queries weren't a good representation of the whole dataset: the top ones barely surpassed 0.5% share. This was in accordance to what was observed in query frequency

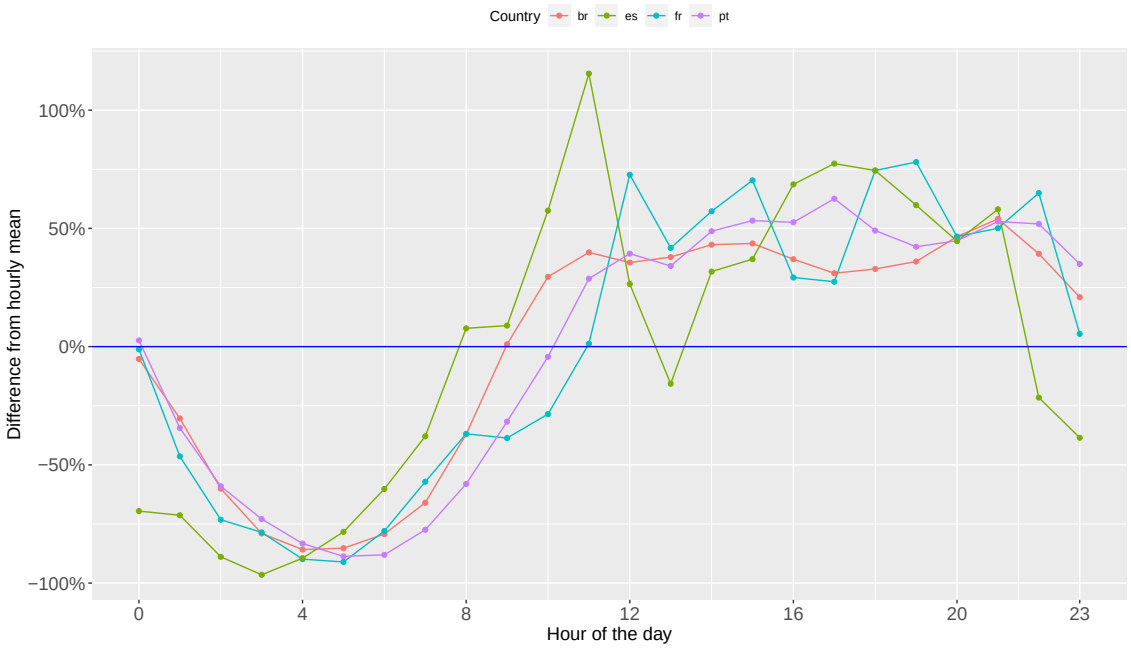


Figure 5.7: Hourly query distribution for the top 4 geolocations (relative difference to mean).

distribution, where we concluded that caching was only effective if we stored a large percentage of the most frequent queries.

5.3.2.1 Behavioral Changes due to COVID-19

The appearance of the COVID-19 and its rapid spread around the world led to a sudden halt in many non-essential industries, which included sports. In Figure 5.5, we saw a first effect of the pandemic on search frequency, with the second of our two weeks of analysis having a consistent drop of 25% or more. Therefore, we isolated log entries concerning this most significant period (March 12th-20th), which contained 129,124 queries, and looked at some metrics that could be subject to behavioral pattern changes. More specifically, we looked at the top queries, geographical distribution and respective hourly distribution for the top geolocations.

The top queries for the period are presented in Table 5.12. Searches now targeted mostly South American (Brazilian) teams, covering more than half of all entries. Moreover, some ambiguous queries could also have been used with the same intent (e.g., [liga] for any competition and [sao] for various teams or players). This was likely a consequence of South American football only coming to a stop more than a week later than European football. Some entities that were already popular overall were even more relevant in this period, maybe explaining why they showed up at all. Moreover, top queries were even less representative, with the top ones converging faster to only just over 0.30% share.

Given the rise in search for Brazilian entities, we analyzed the geographical distribution to assess a corresponding change. The top geolocations were still the same as what we obtained in

Table 5.11: Top 25 queries (overall).

Query	Frequency	Percentage
flamengo	1703	0.53%
benfica		
braga	1385	0.43%
porto	1244	0.38%
sporting	1240	0.38%
santos	1218	0.38%
liga	935	0.29%
botafogo	885	0.27%
vasco	826	0.25%
inter	818	0.25%
gremio	796	0.25%
palmeiras	760	0.23%
pedrinho	700	0.22%
barcelona	695	0.21%
ronaldo	674	0.21%
juventus	670	0.21%
real	662	0.20%
cruzeiro	649	0.20%
sport	621	0.19%
liverpool	598	0.18%
psg	592	0.18%
atletico	569	0.18%
aves	559	0.17%
rio ave	553	0.17%
boca		

Table 5.12: Top 25 queries (March 12th-20th).

Query	Frequency	Percentage
flamengo	634	0.49%
santos	595	0.46%
inter	436	0.34%
sporting	413	0.32%
benfica	408	0.32%
gremio	405	0.31%
botafogo	392	0.30%
vasco	351	0.27%
porto	340	0.26%
liga	329	0.25%
atletico	310	0.24%
sao	303	0.23%
braga	299	0.23%
ronaldo	294	0.23%
sport	287	0.22%
cruzeiro	283	0.22%
palmeiras	254	0.20%
brasil	253	0.20%
real	244	0.19%
fla	233	0.18%
ceara	232	0.18%
barcelona	230	0.18%
arsenal	227	0.18%
bahia	223	0.17%
sao paulo	222	0.17%

Table 5.10 and in the same order. However, there wasn't as clear of a majority as before. While Spanish and French searchers continued to have a similar activity pattern (1.73% and 1.31% share in this period, respectively), Portuguese searchers saw a decline and now represented just 53.82% of all searchers. Moreover, Brazilian originated searches had an increase of over 4 percentage points and amounted to 32.83% of searches, nearly one-third of total.

We also re-observed the hourly patterns for these countries. Results are shown in Figure 5.8. Some shifts were visible. The Spanish, for example, had a delayed lunch time and were more active at night time, possibly due to no outside commitments. Portuguese searchers were active at earlier hours, peaking at 3 PM. Brazilians were now the earliest risers, while the French appeared to be the most consistent group when compared to the overall pattern.

Finally, as a way to directly assess the impact on Portuguese search behavior, Figure 5.9 shows

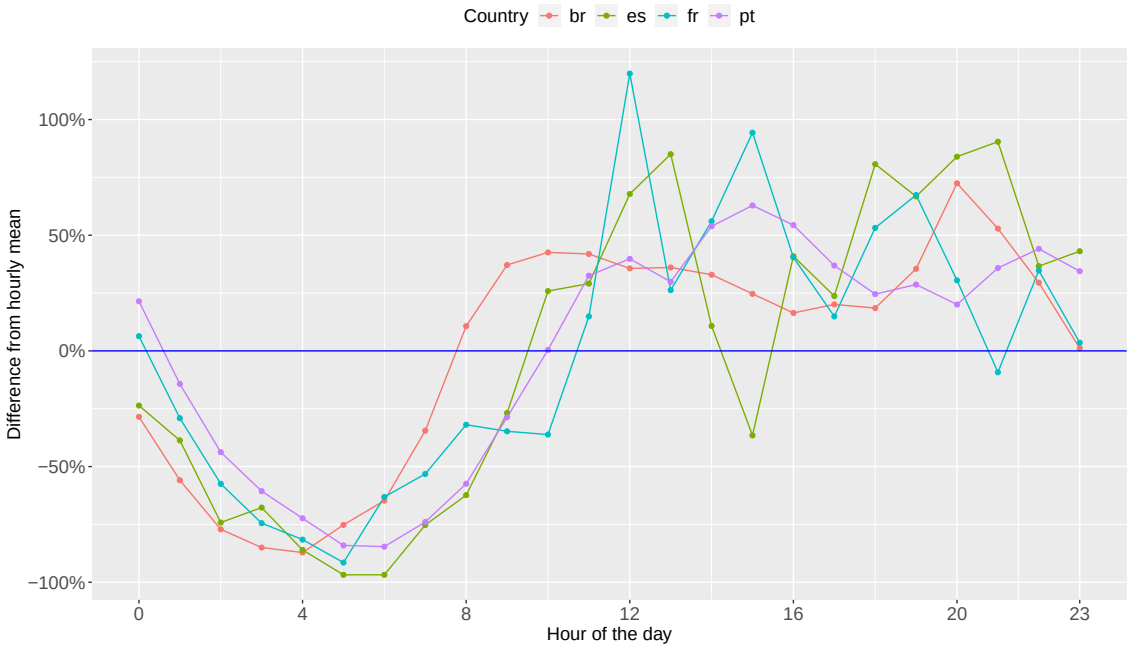


Figure 5.8: Hourly query distribution for the top 4 geolocations in March 12th-20th period.

the hourly activity patterns for searches from Portuguese users in the first (pre-COVID) and second (post-COVID) weeks of the collected period. Notice that percentage differences shown are relative to each subset's average for their respective period, i.e., the Post COVID line's average is relative to the mean in the 12-20th March period, while the Pre-COVID line's mean is calculated from the preceding period.

5.3.2.2 Search Behavior Surrounding Football Events

Every engine is subject to spike events that temporarily alter normal user search behavior. We did not go into as much as detail as some related work, like Mansouri et al. [56], but used some of their ideas. More specifically, we picked a relevant event during our collection period and analyzed querying behavior pre, during and post the event's time frame. The event could have been a competition drawing, an unexpected sports related incident, or just a game that generated activity. We chose the latter, as it was the only option available. Therefore, we looked at the game FC Porto 1-1 Rio Ave[3], which took place on March 7th, at 8:30 PM. For this, we isolated log entries from that weekend so that we could, first, assess some metrics for this subset, as a way to verify if it was an acceptable approximation of the full dataset.

The top two queries for the period were [porto] and [benfica]. Given the matches that took place and respective circumstances (both teams played between 6 PM and 10 PM and were in contention for the top league position), this was expected. Next most frequent queries included [flamengo], [braga], [liga] and [sporting], showing that, despite some order change, frequent

[3] https://www.zerozero.pt/match.php?id=6958726 (accessed April 2020)

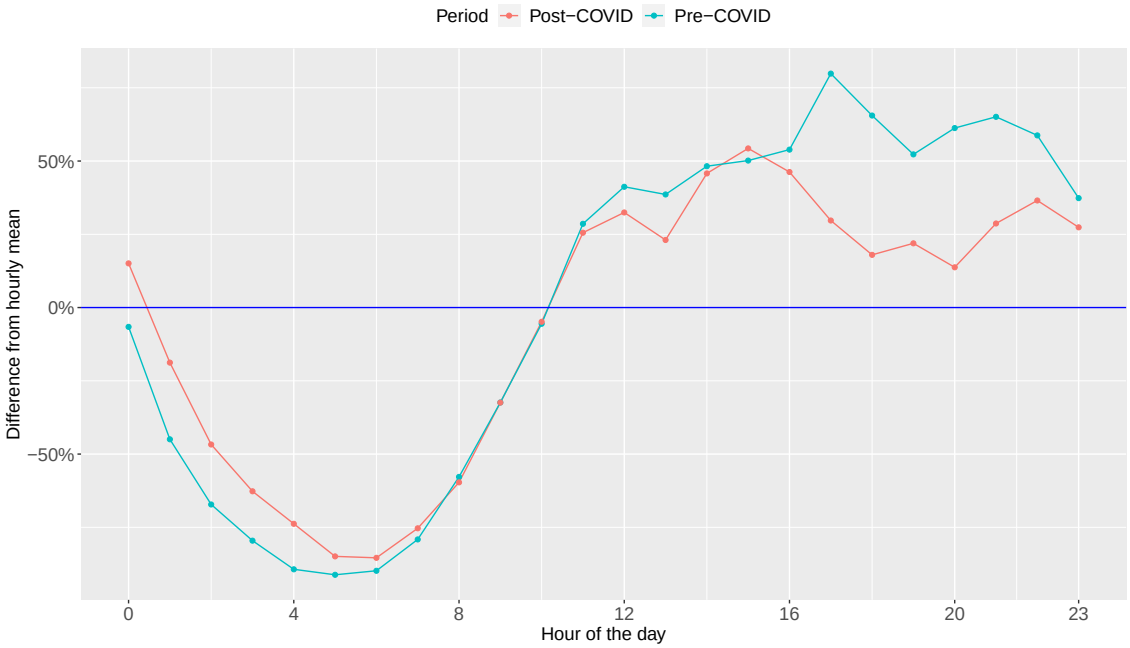


Figure 5.9: Hourly query distribution for Portuguese searchers (pre vs post COVID).

queries were reasonably constant in that period. Queries were, however, more diverse. Compared to the statistics presented in Table 5.6, in this period, the ratio of unique queries rose to 31.19%, with over 19% of queries occurring only once. Moreover, the caching effect was less effective: storing 0.4% of top queries would only cover 15.2% of queries. By storing the top 20%, the value would achieve 68.84% coverage. A pattern that remained constant was query length and structure. One term queries still accounted for over 70% of all occurrences and the average sat at 1.39 terms, close to the original value. We then saw that some patterns were constant, while others seemed more mutable according to the period.

We identified the three main periods concerning pre, during and post game. The first covered the 90 minutes prior to the starting time, the second 2 hours from that point, and, finally, the last the remaining of the day (2 more hours). For each, we looked at the top queries submitted, as well as analyzed a query minute distribution in 5 minute bins. Here, it was no longer necessary to segregate by location and adjust timestamps, as we were interested on covering each period in Portuguese time anyway. Table 5.13a and Figure 5.10 show results obtained for the pre-game period.

Most top queries concerned the game, though indirectly. With a Benfica game[4] starting earlier and ending at around 8 PM, the bigger activity period in the 90 min interval, most searches pointed to the entities involved there. Pizzi[5], one of the main Benfica players, had missed a penalty that could have won the game and secured first place. Setúbal were their opponents (other expressions were also present that indicate people searching for the team, e.g., the 13th and 14th top queries

[4] https://www.zerozero.pt/match.php?id=6958720 (accessed April 2020)
[5] https://www.zerozero.pt/player.php?id=51004 (accessed April 2020)

Table 5.13: Top queries around game period.

(a) Pre-game.

Query (Freq.)
benfica (58)
pizzi (32)
setubal (27)
liga (25)
porto (22)
flamengo (17)
audax (17)
rio ave (16)
thomas muller (14)
barcelona (14)

(b) During game.

Query (Freq.)
porto (48)
rio (43)
benfica (35)
taremi (27)
rio ave (27)
flamengo (26)
liga (17)
sporting (16)
fla (16)
braga (16)

(c) Post-game.

Query (Freq.)
flamengo (45)
porto (36)
liga (36)
benfica (30)
santos (21)
rio ave (18)
fla (15)
braga (14)
audax (13)
toronto (12)

were [setúbal] and [setu], respectively). Moreover, we also saw some searches regarding the upcoming match, with both teams' names appearing in the list. Nearing the beginning of the match, query frequency dropped, a sign that an important game was going to start.

After entering the game period, there was a behavioral change. Table 5.13b and Figure 5.11 show the results. Naturally, there was a decline in the number of queries compared to the pre-game period, for people were more invested in viewing. Moreover, query frequency rises could be almost perfectly matched to three key periods. The first, at around 8:55 PM, was near both teams' goals. This was also reflected in the top queries, where Rio Ave's scorer, Taremi[6] appeared in 4th place. Despite not being present in the top 10, another player with a good exhibition in the match, Nuno Santos[7], appeared in 12th place, with 15 occurrences. The following moments saw bigger average activity with another spike close to an hour after the game start, corresponding to half time. Finally, there was another, more noticeable spike around 10:05 PM, in the end of the game, likely helped by the unexpected outcome (draw) and its implications in league standings.

Last, in the post-game period, activity did not regain the values it had pre-game. Table 5.13c and Figure 5.12 show the respective results. There was an initial period, of about 30 minutes after the game was completed, where users still searched more frequently. They were likely attempts at analyzing the implications in the league's standings after these results (notice that [liga] was the tied 2nd most frequent query). After that, typical night time activity was quickly resumed.

We then inferred that game-like events lead to very localized spikes and search behavior changes whose effects were not felt much outside the game period. It would be interesting to assess if this pattern would be recurring amongst other types of events (e.g., a big competition draw or important award ceremonies), or if their span varied. Unfortunately, we were unable to collect logs that covered a wider range of events to allow this analysis.

[6] https://www.zerozero.pt/player.php?id=408631 (accessed April 2020)
[7] https://www.zerozero.pt/player.php?id=160873 (accessed April 2020)

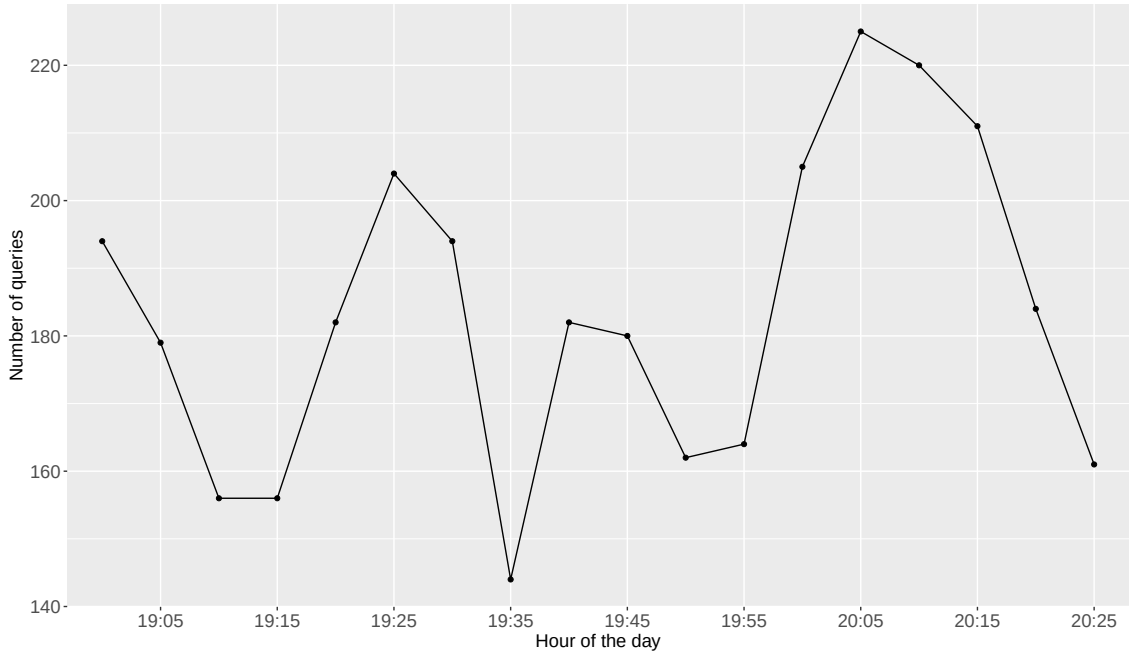


Figure 5.10: Pre-game query minute distribution.

5.3.3 Session Level

As we previously defined, a session is a set of interactions, performed by the same user, with each interaction being separated by a cut-off inferior to some threshold. We defined the threshold as 30 minutes, as this was a common value observed in other studies [21, 80, 82, 64]. In total, there were 190,980 sessions across the dataset.

One of the used metrics to evaluate session behavior is its length in number of queries. This metric demanded only unique query entries from the log and was the only one, at session level, that applied this filter described earlier, in the section's introduction. The mean number of queries per session was 1.87. Table 5.14 shows the full values distribution.

Results showed that in around two-thirds of sessions considered, users performed only one query. This was also observable in previously obtained results, be it domain-specific or not, and even if at different scales. For the AltaVista study [69], for example, one query sessions had a 77.6% share, although, due to their 5 minute cut-off, this value might have been higher than reality. From 10 queries onwards, values got lower and lower, with entries of 17 queries and higher coming up to 0.05% or less each. The maximum value registered was 93 queries, which only happened once. In this session, the user was authenticated, and performed several distinct queries for mainly Brazilian entities, with no query having over 3 occurrences.

The mean number of queries also did not deviate much from other results. Our value was very close to the 1.9 of open Web search engine Excite [74] and expert search engine INDURE [31], and not far off the 1.73 of Chau et al. [21]. The latter offers more guarantees of similarity, as it used a temporal cut-off with the same value. These results indicate that, no matter the context, either users were not eager to refine their searches, or found their results very easily.

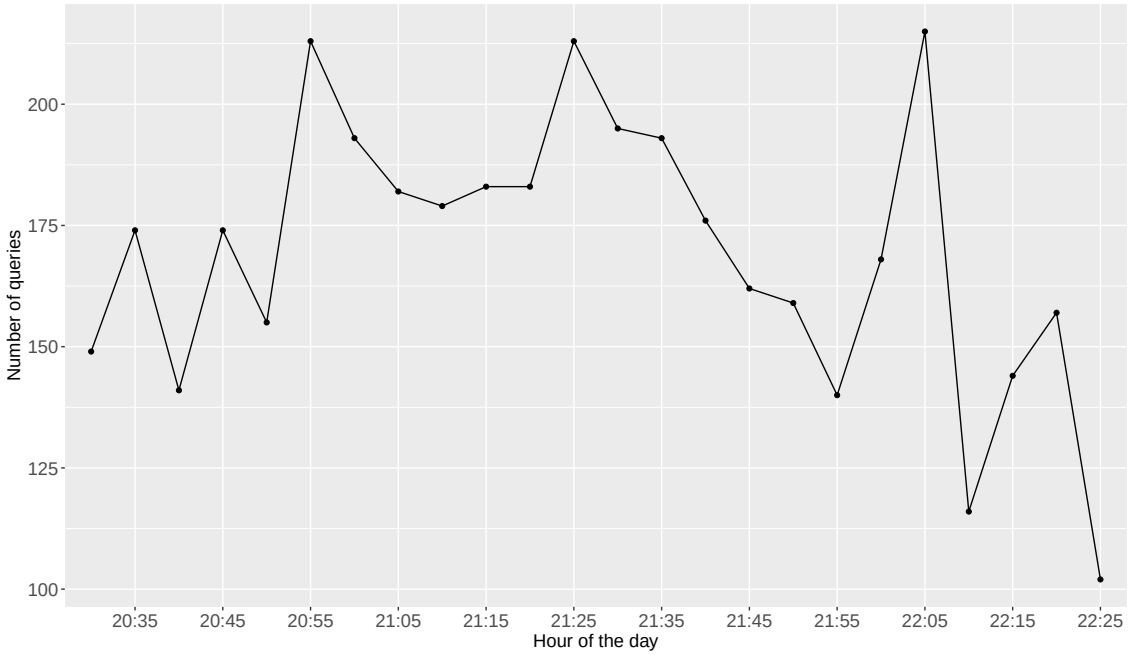


Figure 5.11: In-game query minute distribution.

Session duration could further help understand this pattern. A session's duration is defined as the difference between the timestamps of the first and last recorded interactions. As we had result clicks, our durations were likely to be more accurate than datasets that only registered queries, as the result navigation component in those was scrapped. However, the cut-off value assumes most importance and influence, and comparisons should be carefully made. The mean session duration was 3 minutes and 18 seconds. Table 5.15 shows the distribution of session duration in time bins.

Results showed that most sessions did not last longer than 1 minute. This followed from our previous result: as most sessions had only one query, unless the user spent a long time scrolling through results, the interaction with the system was going to be fast. There was a decline in frequency as the duration increased, with a noticeable exception in the range of 15 to 30 minutes. Janssen et al. [42] reported that the average Web session lasted around 15 minutes. Although not the case here, it was interesting to observe this little spike. Perhaps some search patterns do last throughout time, in one way or another. Sessions longer than our cut-off between consecutive interactions, i.e., 30 minutes, amounted to less than 3% of the total. Torres et al. [80] did not provide exact values, but their result plots hinted at a value of a little over 60% of sessions lasting up until 10 minutes in the context of IR for children. The mean duration value of 5 minutes and 23 seconds for other Portuguese studies [64] was higher than our result. For Fang et al. [31], this value was even higher, at 18 minutes and 40 seconds. However, the latter had no temporal cut-off, which may have biased their result. Moreover, as the engine targeted in our study demands less analysis of the results' potential ability to satisfy the underlying information need, we found our lower mean to be within the expected result range.

Finally, we looked at how users paginate through results throughout a searching session. We

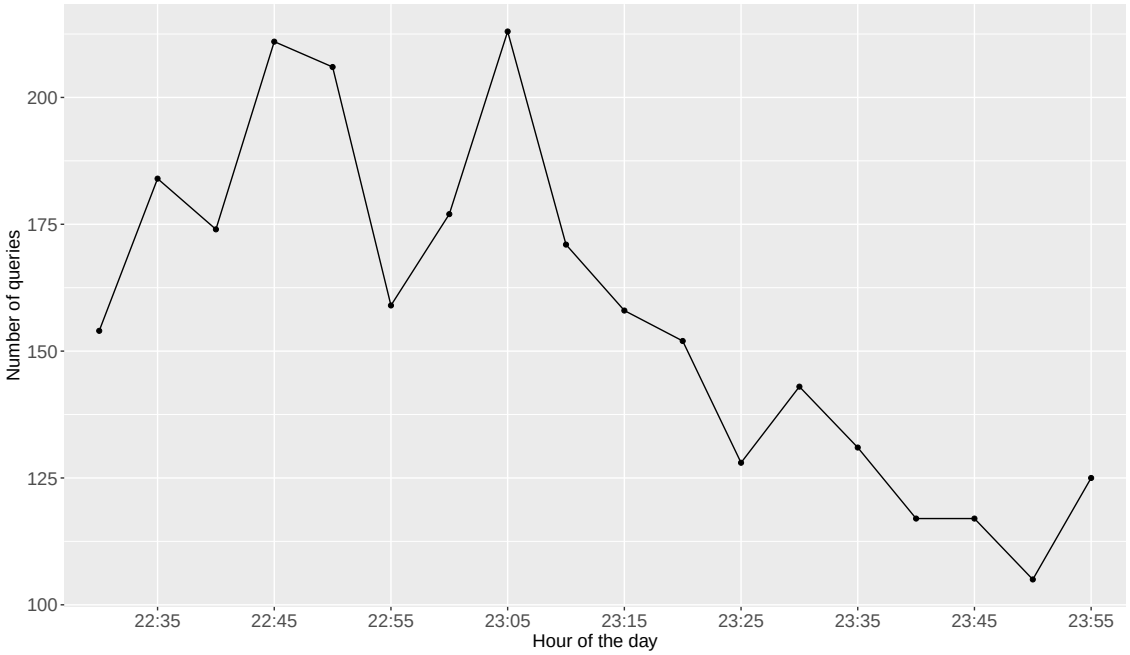


Figure 5.12: Post-game query minute distribution.

were most interested in the average number of result pages visited throughout a session, as well as the concrete page distribution, regardless of the search made. For the former, we obtained a value of 1.04 pages. This skewness was somewhat unusual: in Excite studies, only in 2001 was one page the majority, with just over 50%. Furthermore, Chau et al. [21] arrived at a value of 1.47 pages. One possible explanation for this value may lie within the context of the engine analyzed. Like in enterprise search, users are usually precise on what they want to find, and there are few, or even a single result that satisfies their need, that is expected to be (near) the top entries returned. This happens because documents are entities with little free text, hence searches are expected to have extreme precision. Table 5.16 shows the distribution for the result pages visited throughout a session. Note that interactions on the main page search popup were considered to be first page, as their results were identical to the real ones, were the user to get there. Moreover, in one session, multiple pages could be visited.

Values for pages 10 and higher amounted to a notable percentage, considering the frequency of other pages with the exception of the first one. However, most values in it occurred once, and several were in the hundreds or higher range. As it is unlikely that one would scroll manually to those pages, it seems that these corresponded to desperate attempts of users at finding the desired result result, or other kind of behavior that contributes to this output (e.g., undetected bot activity).

5.3.4 Click Level

As described, to perform a click level analysis, we isolated the entries from our post-preparation log that had a positive value for click ranking position. This accounted for 270,496 entries. Over 80% corresponded to clicks on suggestions in the main page search bar. However, this value could

Table 5.14: Session length distribution.

Queries	Sessions	Percentage
1	116,315	67.05%
2	30,129	17.37%
3	11,362	6.55%
4	5,622	3.24%
5	2,987	1.72%
6	1,887	1.09%
7	1,179	0.68%
8	803	0.46%
9	587	0.34%
10+	2,597	1.50%

Table 5.15: Session duration distribution.

Duration (minutes)	Sessions	Percentage
[0, 1[	145,455	76.16%
[1, 5[	19,706	10.32%
[5, 10[	7,872	4.12%
[10, 15[	4,600	2.41%
[15, 30[	8,316	4.35%
[30, 60[	3,645	1.91%
[60, 120[	1,150	0.60%
[120, 180[	188	0.10%
180+	47	0.03%

be somewhat skewed. The reason for this is the decision of the search engine to, sometimes, automatically redirect users to the page corresponding to what would have been the first result shown. In these cases, a click ranking position of 1 is registered. More often than not, this might be what the user was searching for, though we can not guarantee that was their original intention.

Click level analysis is not common and, therefore, it was hard to compare results in order to establish differences or similarities with other domains. Nevertheless, we present what we think are a set of metrics that are easily implementable and allow for an interesting analysis per domain, while also providing values for future benchmarks.

We started by analyzing the main entity types that were clicked by users as a result of their searches. Players and teams were the most clicked type of results, with approximately 50.2% and 32% of all clicks, respectively. This did not come as a surprise, since these two entities are the main focus, not only on football, but team sports in general. Competitions and managers completed the top 4, with 7.2% and 2%, respectively, for a combined total of over 90% of all click entries. Competitions are where teams face off and the main gateway to accessing past, present or future encounters between them, as matches were not directly searchable within the search engine (otherwise, due to their bigger presence, they could heavily bias the results list). Managers, being responsible for teams and often one of the main images of a club as well, also instigated curiosity in searchers. Other clicked entities that were less accessed included agents, directors, places, stadiums and referees.

Regarding the overall click distribution per ranking position, as expected, the majority lied within the top results. The ranking was calculated considering page number, e.g., if a result was clicked on the 10th position of page 3, then its rank would be 30. Results followed from the pagination distribution performed during session analysis, where, on average, users visited around one page per session, i.e., didn't move past the top 10 results. While there were cases where some users went to higher page numbers, rarely did those result in actual clicks. The lowest ranking that resulted in a click was a 10th position result in the 93rd page, for a query [victor eduardo ramos].

Table 5.16: Result page visit distribution by sessions.

Page number	Sessions	Percentage
1	190,674	96.04%
2	2,899	1.46%
3	1,425	0.72%
4	874	0.44%
5	400	0.20%
6	285	0.14%
7	132	0.07%
8	115	0.06%
9	83	0.04%
10 and others	1651	0.83%

Table 5.17 and Figure 5.13 detail the overall click distribution. Top results had a 72.65% majority in the distribution. In contrast, Torres et al. [80], while studying how children differed from adults on the search task, arrived at a result of about 60% of click ranks associated with the first result for adults, and a little over 40% for children.

Table 5.17: Overall click ranking distribution.

Ranking	Clicks	Percentage
1	196,517	72.65%
2	35,187	13.01%
3	13,905	5.14%
4	8,066	2.98%
5	5,359	1.98%
6	3,572	1.32%
7	2,565	0.95%
8	2,161	0.80%
9	1,358	0.50%
10	1,209	0.45%
11+	597	0.22%

We partitioned this distribution from different perspectives to assess consistency. Considering only clicks on the main page search bar, results were almost equal: the top 3 results had a share of 72.34%, 13.57% and 5.32%, respectively. Since the results suggested when typing are the same as shown in the first full results page, this did not come as a surprise. Moreover, we analyzed the distribution per entity type. For the four major entities identified previously, the top result always had a share of over 50%. However, despite the distribution pattern similarity in the case of players (73.02% for the top result) and teams (80.97%), the value worsened in the case of competitions

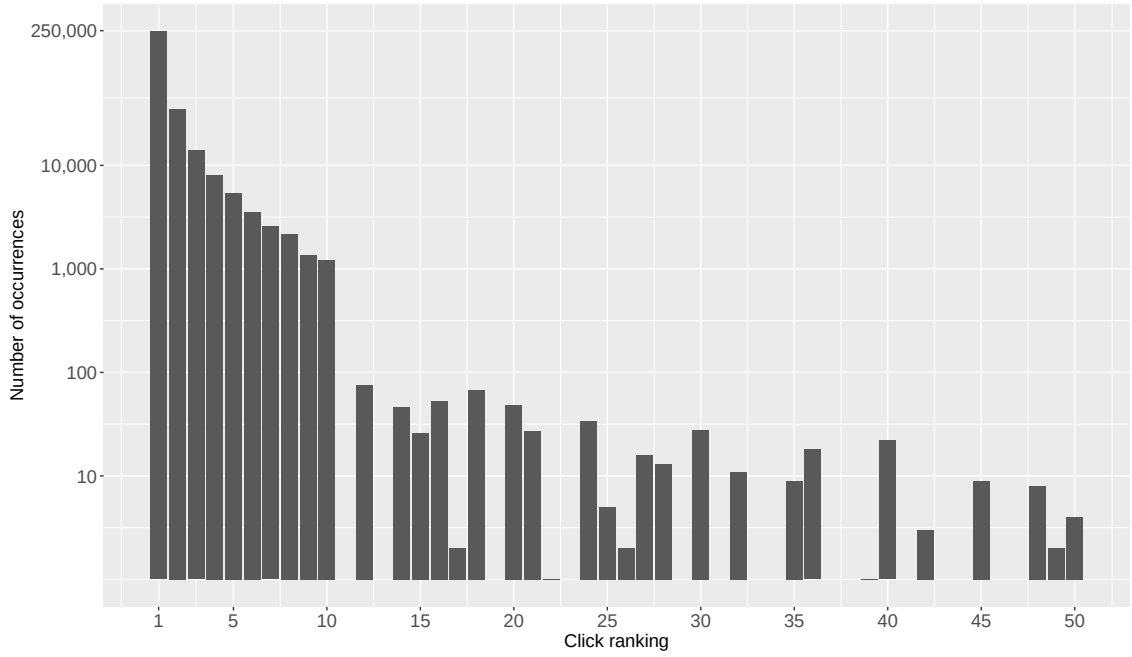


Figure 5.13: Full click ranking values distribution (log-linear scale).

(50.59%) and managers (63.88%).

We also looked at click behavior change throughout a user session. For this, we collected the last and maximum clicked rank positions on each session, in order to assess how effective users were in translating their information needs to the system after a possible query reformulation. With that information, we calculated the ratio of sessions that saw an improvement in clicked rank position, i.e., the last clicked result was the highest ranked one. We found that 71.91% of sessions saw an improvement. Sessions with only one clicked result were discarded, since they would bias this result towards a higher value. When including them, 79.58% of sessions ended with a click on a top result.

Finally, we aimed at listing queries by some criteria regarding associated clicks. For that, first we looked at click entropy. Though some users may have searched using the same query, the underlying information need could vary, i.e., there was different intent. An example could be the query [liga], that can refer to several competitions. This is likely common with smaller queries, for their lack of specificity. With this in mind, Kulkarni et al. [51] proposed an analysis on query intent variability by using the concept of entropy. They defined the entropy for a query q as:

$$Entropy(q) = -\sum_{u \in P_c(q)} p_c(u|q) \times \log p_c(u|q) \tag{5.1}$$

where $P_c(q)$ is the set of entities clicked when searching using query q and $p_c(u|q)$ the click share of entity u for query q. We replicated this measurement in our analysis. However, we took multiple perspectives. To assess entropy at a coarser granularity level, we also calculated entropy at entity type level (e.g., team, player, manager). Tables 5.18a and 5.18b show the results for query entropy

at individual entity and entity type levels, respectively. Figure 5.14 shows the full distribution of entropy values for individual entities. Note that only values for queries with more than one click were plotted, as all other cases had, by default, no entropy.

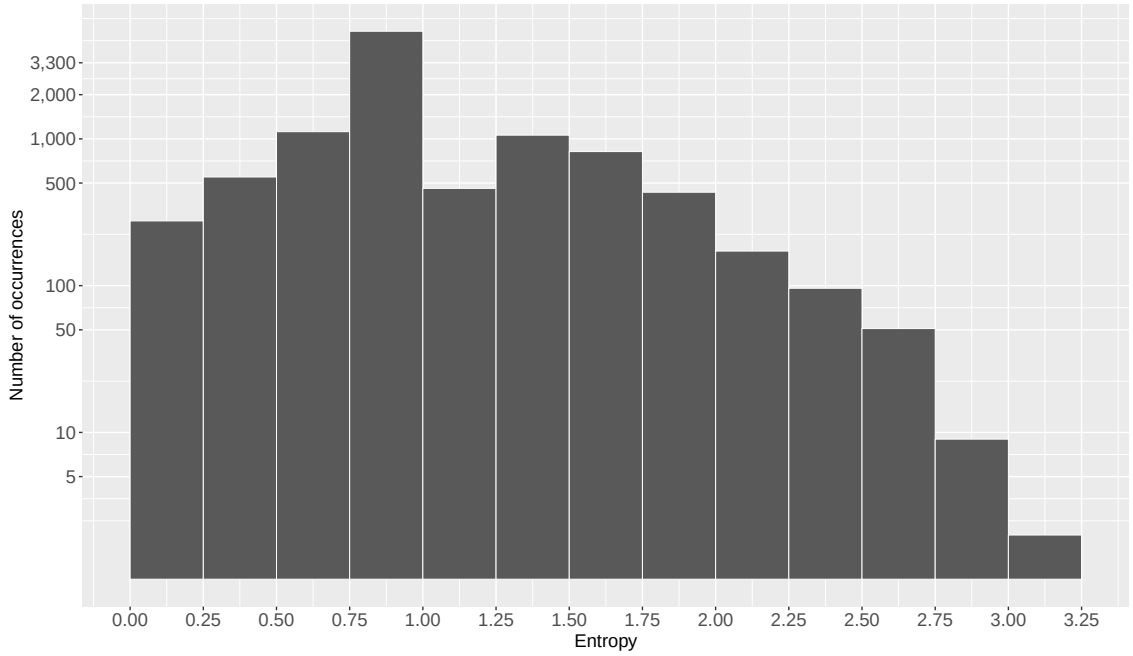


Figure 5.14: Entropy values distribution (individual entity level; log-linear scale).

As expected, queries that lead to a bigger variability in clicks were shorter ones, rarely having more than one term. Moreover, some queries appeared to be incomplete. This likely derived from the suggestion mechanism of the main search bar that suggested results when the user typed in at least 3 characters. An example was the query [marc], which could expand to [marco], [marcelo], [marcos], or even really just be the first name Marc. Most top queries with higher entropy at entity level concerned these personal names and were expected to lead to a variety of different results. The query with most different entities clicked on this top was [diogo], followed by [nacional], which could refer to teams (e.g., Portuguese team Nacional da Madeira[8]) or numerous competitions. At entity type level, values were, naturally, lower, as there were only close to a dozen (11) possible entity types. Here we saw a tendency of place-like names (e.g., [maracana], [luxemburgo], [marino]). City names normally refer to the respective place, but are also often present in team and stadium names, and less often as people surnames.

This inherent variability associated with the search context meant that it was difficult to use entropy as a query retrieval quality scoring mechanism, i.e., a metric to evaluate the satisfaction of the query's results to the user who typed it. Furthermore, entropy only measures result variability, not taking into account the positions in the result set, a key factor in how well a potentially relevant was highlighted to the user. For this, we wanted to use click ranking information. After initially

[8] https://www.zerozero.pt/equipa.php?id=27 (accessed March 2020)

Table 5.18: 20 queries with highest entropy.

(a) Individual entity level.

Query	Entropy	#Entities
luis	3.22	11
pedro	3.14	11
marc	2.99	9
wellington	2.95	9
saulo	2.92	8
cesinha	2.86	8
paulao	2.85	8
jorginho	2.81	9
ward	2.81	7
paraguay	2.77	10
nacional	2.76	13
jose semedo	2.75	7
brasileiro	2.74	10
balde	2.73	7
diogo	2.73	15
silv	2.73	7
cesar	2.72	7
flavio	2.72	7
aguas	2.72	8
emerson	2.71	8

(b) Entity type level.

Query	Entropy	#Entity Types
maracana	2.00	5
sala	1.92	4
latte	1.92	4
lux	1.91	4
luxemburgo	1.90	4
marino	1.89	4
isarel	1.89	4
neca	1.88	4
gouveia	1.86	4
pochettino	1.86	4
capi	1.84	4
tiago ferreira	1.84	4
argel	1.80	4
carraca	1.79	4
rui borges	1.79	4
congo	1.79	4
matosinhos	1.78	4
franca	1.76	4
rodolfo reis	1.75	4
madeira	1.74	4

trying to unsuccessfully order queries based on average ranking, we hit an obstacle due to the bias of low frequency, medium to high rank click queries. To avoid this, we once again adapted the traditional TF-IDF weighting scheme [55, Chapter 6]. Therefore, we proposed that a query q's retrieval fulfillment score (RFScore) be calculated from the following equation:

$$RFScore(q) = \sum_{r \in CR(q)} Freq(q,r) \times \log_{10} \frac{N}{r} \tag{5.2}$$

where $CR(q)$ is the set of ranking positions where q led to a click, $Freq(q,r)$ is the amount of times the query led to a click on that position, and N is a scaling factor to favor higher rankings (i.e., better matches). In traditional TF-IDF, it would be the collection size, as it represents an upper bound on a term's document frequency. Hence, we used the lowest ranking position clicked, which was 930. With this formula, we could favor queries with more clicks on higher positions and vice-versa. Tables 5.19 and 5.20 show the top 10 best and worst scored queries, respectively. Table 5.21 shows the worst scored queries with at least 5 clicks.

Most queries with the best score coincided with the most frequent queries submitted to the

Table 5.19: 10 best scored queries.

Query	Score	#Clicks
sporting	5141	1742
flamengo	4480	1514
benfica	3815	1296
santos	3682	1316
braga	3176	1087
porto	2973	1010
inter	2568	906
vasco	2412	814
botafogo	2397	842
liga	2374	889

Table 5.20: 10 worst scored queries.

Query	Score	#Clicks
campeonato equador 1a liga 2020	0.291	1
ramon menezes defensor	0.477	1
liga polonia	0.662	2
sporting clube vila verde	0.947	1
victor eduardo ramos	0.987	5
w. montoya	1.070	1
campeonato suico	1.123	1
luiz felippe	1.190	1
lobato	1.190	1
liga bulgaria	1.220	1

system. Indeed, overall they were single terms that referred, more often than not, to a popular entity, or a very restrict set of entities, that were easily searchable. Note that some ambiguous queries, like [liga], were also well ranked. While it is unlikely that clicks were always in the top result (as there are many competitions to search for), usually users looked for ones that appeared in the top results. This shows the impact of a ranking's frequency in our formula: even if not on the top, if frequent and high enough, it could still be well-scored.

As a complement, we plotted the scores for the top 100 most frequently submitted queries. Figure 5.15 shows how their scores evolved. Despite the overall downwards tendency, it was not a linear relationship between query frequency and score: some less frequent queries had higher scores due to the discriminative IDF factor, suggesting clicks in typically higher result rankings.

Table 5.21: 10 worst scored queries (minimum 5 clicks).

Query	Score	#Clicks
victor eduardo ramos	0.987	5
liga chinesa	7.550	5
lokomotiv minsk	8.586	5
distrital guarda	8.941	5
garcia	9.306	5
chris philips	9.526	5
sporting clube de lourel	9.588	10
monte caparica	9.842	5
van der sar	9.980	5
as roma	10.122	5

On the other end of the spectrum, the worst-scored queries were ones that, ultimately, weren't as frequent, and resulted in low-position clicks when present. It was interesting to note the pres-

ence of two entries that represented a specialization of the ambiguous and high-scored query [liga], but resulted in poorer results. This could be a sign that the current engine values some general terms too much and does not consider other specialized ones. On a manual experiment, we entered the query [liga bulgaria] on the search engine and found the relevant result only in the 38th overall position.

Finally, when setting a minimum click threshold, some score differences were visible, likely due to the higher number of entries associated. However, queries were equally longer, suggesting that the current system can't always provide the same type of satisfaction when providing answers for these cases.

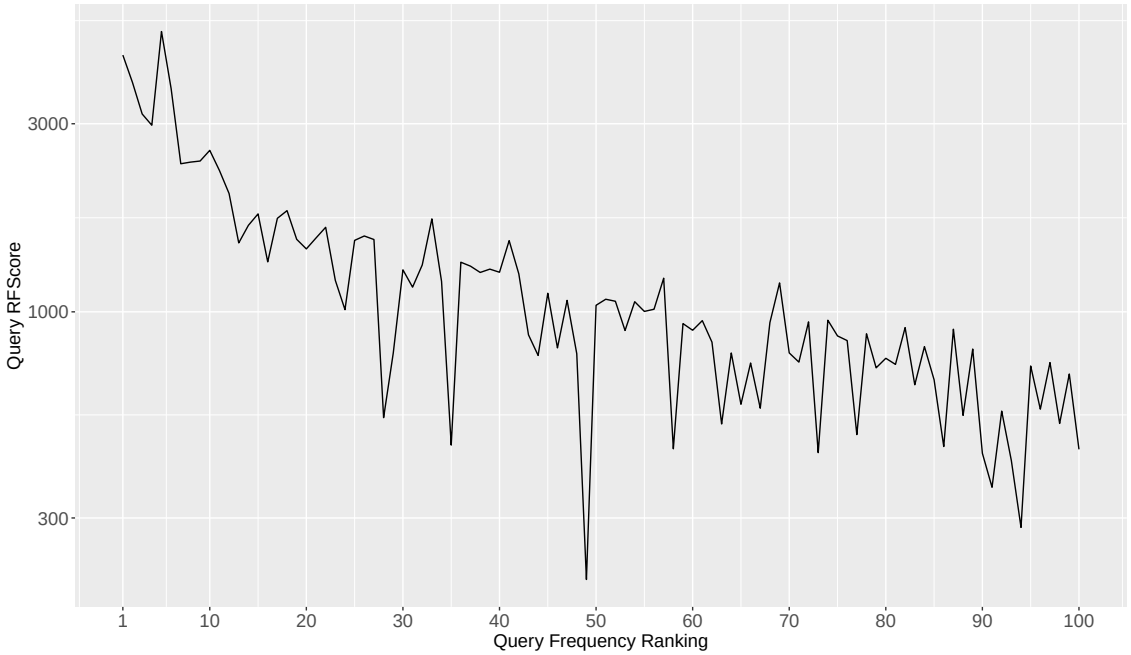


Figure 5.15: Query RFScore evolution for 100 most frequent queries.

5.4 Summary

Analysis of the zerozero search logs through a QLA process allowed us to better understand how users in this particular domain interacted with the search engine. Moreover, we were able to detect consistencies and differences in distinct metrics when compared to previous studies concerning several types of domain.

We started by preprocessing the dataset, eliminating machine-originated queries by filtering suspicious User-Agent strings and unusually long sessions. The number of filtered results was lower than other studies that were known to have applied similar mechanisms, though most were open Web engines and, therefore, subject to bigger activity, human or not.

We then analyzed search behavior from four different perspectives: term, query and session, the most commonly used, and click, rarer due to the difficulty in collecting such information. Query terms were found not to be long, averaging 6.44 characters. The full values distribution resembled a normal curve, even when considering outlier terms with higher character counts. Like other studies, unique term percentage (10.72%) turned out to be context specific and different from values of other researchers. Moreover, top terms provided little representation of the whole dataset. Top terms included words that could be perceived as full queries. Query bigram analysis revealed that they weren't nearly as frequent as single terms, which was expected.

Query analysis confirmed patterns observed in the term level perspective, namely an average caching effect and short lengths, with single term queries assuming bigger presence. The analysis of new searches showed that a little over 20% of queries submitted each day had never been seen before. Moreover, users tended not to radically change their initial query attempts. When doing so, modifications were small, i.e., at most one term at a time, more often that not. Portuguese searchers represented close to 60% of the user base, which overall preferred to search on weekends. However, query volume suffered a drop midway through the collection period, due to the COVID-19 pandemic's effect on sports events. This led to some behavioral changes, more specifically on hourly patterns and what entities users searched. Finally, we observed how football matches caused localized search spikes that didn't span much beyond the timeframe of the event.

Search sessions were found to have typical Web patterns, being short both in length (average of 1.87 queries) and duration (a little over 3 minutes). Moreover, pagination analysis showed that an overwhelming majority of searchers did not go past the first results page in pursuit of a correct answer. In reality, though, the value may have been skewed, due to the auto-redirecting mechanism currently in place in the production search engine.

Finally, at click level, with little benchmark for comparison, we observed that most clicks lied, unsurprisingly, on top results, with frequency following a power law for that subset. Furthermore, query click entropy analysis showed that shorter queries, often single names, were the most ambiguous. This pattern was also observed when measuring entropy at entity type, a coarser granularity. Last, we proposed a query fulfillment metric based on traditional TF-IDF formulation that showed how longer queries produced lower quality results for users in the current engine.

Chapter 6

Search Engine Implementation

With an understanding of where this particular search task lied and how users searched, we can propose an enhanced search engine that helps identify relevant results for a larger set of queries. However, before detailing its implementation, and since we are in a cooperative environment with full access to the target collections, it is important to understand the data in them, so that we can further fine tune the retrieval process. After this, we describe the choices we made in order to build the new engine that will retrieve the analyzed data.

It is relevant to note that, for security reasons, we will describe and further work not with the full production collections, but with samples instead. These samples were the product of a partial database dump and were provided with the remark of being a good representation of the overall entities present in each collection by the zerozero.pt team.

6.1 Collections Description

In order to better understand the information at hand, we analyzed some of the main entities present and searchable within zerozero.pt's search engine. Due to the system growth since the project's inception, all entities were populated with a large number of fields that served different purposes beyond entity identification and description (e.g., fields concerning their internal status, creation and last edition information, and image paths and configurations for display). Therefore, we filtered a small selection of what were considered the most relevant fields that best help identify an entity or contain information that was more likely to be used by potential searchers. Moreover, we focused on the four main entities that were most searched by users, according to previous click analysis in Chapter 5, and had bigger presence in the system database: Competitions, Teams, Managers and Players. Due to working with a sample, some analysis results might not always be expected. We single out these situations appropriately.

6.1.1 Competition

A Competition represented some tournament in which a number of teams participated and that ended with one winning team. The competition format varied, as well as their scope and teams.

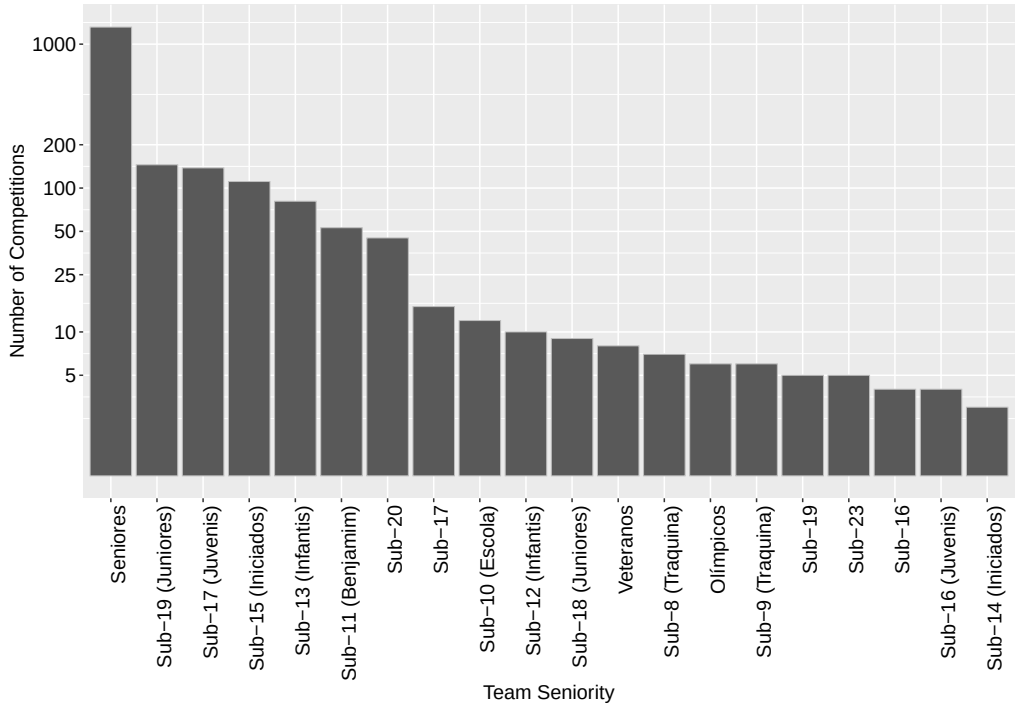


Figure 6.1: Participating teams type for Competition collection sample.

Our sample contained 2,000 competitions. Each one had a name stored in a description-like field. This was a longer name that usually appeared in the navigation menus available on the website. Furthermore, there was an abbreviated version stored that could, or not, differ from the full name. While it wasn't uncommon for both fields to match, sometimes the latter presented an alternative designation more familiar and universal (e.g., the Spanish league stores "Liga Espanhola" as their description, but "La Liga", a more universally recognised name, as abbreviation). Regarding the teams involved, there could be a distinction at different player age level that better helped discriminate what users were looking for. One field pointed to information on what kind of teams were involved in the competition, i.e., their seniority. Figure 6.1 shows the distribution of values for this field.

Despite most competitions being at senior level, there were also some at lower levels. Since they could have different designations (e.g., Sub-19 are also called "juniors" and "juniors A"; for Sub-17, the terms can also be "juveniles" and "juniors B"), the importance of this information transcended characterization. Finally, a textual search field, common to some of the other entities, stored a comma-separated list of expressions that users may have been likely to search for. However, in this sample, only 0.8% of rows had a non-empty value.

Last, there were some enumeration fields that further helped contextualize the competition. A team type field indicated if it was a club or national team competition, with over 90% of sample entries concerning club competitions. A scope field indicated whether the competition occurred at worldwide level, continental, national, district or even municipality. There was also a competition

format field (to differentiate leagues, cups and different types in each one), with a little over two-thirds of competition samples corresponding to national leagues. While it could be useful for search, most competitions had their type integrated into their name already. Finally, a season field specified the type of time period the competition occurred at (e.g., there are competitions that only spread one calendar year, common in South American countries, while others start in one, but continue across another, as happens throughout European countries). The latter was most common, with over half the competition samples occurring that way, which was expected given the larger number of European countries and football associations.

6.1.2 Team

A Team is the entity that participates in competitions, playing against other similar entities. It is composed of several players and overseen by a manager. As observed in the Competition sample, teams could be aggregated according to age group, representing different stages of development throughout a typical career.

Each team in our collected sample of 5,000 entries had a unique name stored in a dedicated field. There was also an abbreviated version stored (e.g., Portuguese team Benfica had "Sport Lisboa e Benfica" stored as the full name, but simply "Benfica" as an abbreviation). However, teams are not always addressed by their name, abbreviated or not. Two common designations are nicknames and initials. For example, Benfica is often referred to as "SLB" (initials of their full name), as well as "Águias" ou "Encarnados", due to their mascot and main kit color, respectively. These alternative designations were not always present: in the sample, only two-thirds of teams had at least one nickname. However, given this information, we were equipped with an important vocabulary extension.

Type-wise, there was the distinction of different team types according to the level they played at (e.g., Benfica main team vs Benfica's sub-19). On our sample, most teams corresponded to senior levels. Figure 6.2 shows the full values distribution. While a majority was expected, there seemed to be a lack of teams of lower levels, despite developmental being more and more present in the modern game. From the search pattern analysis performed in Chapter 5, we saw that, while not being amongst the most frequent queries, these types of terms often made a presence (e.g., Table 5.5 contained the bigram [sub 19]).

Finally, there were some fields that better helped describe a team. One of them stored the team's origin city. This could be helpful in eliminating some ambiguities (e.g., when searching for Sporting, we could be looking for Sporting Lisbon or Sporting Braga), although often the team city is incorporated into the name. There was also a field with older names that the team went by in the past. Despite being present in only 5% of sample entries, it could help potentiate search matches for older users, or just younger ones with curiosity. Last, there were fields that helped characterize a team, but were expected to be seldom used in searches (e.g., past achievements).

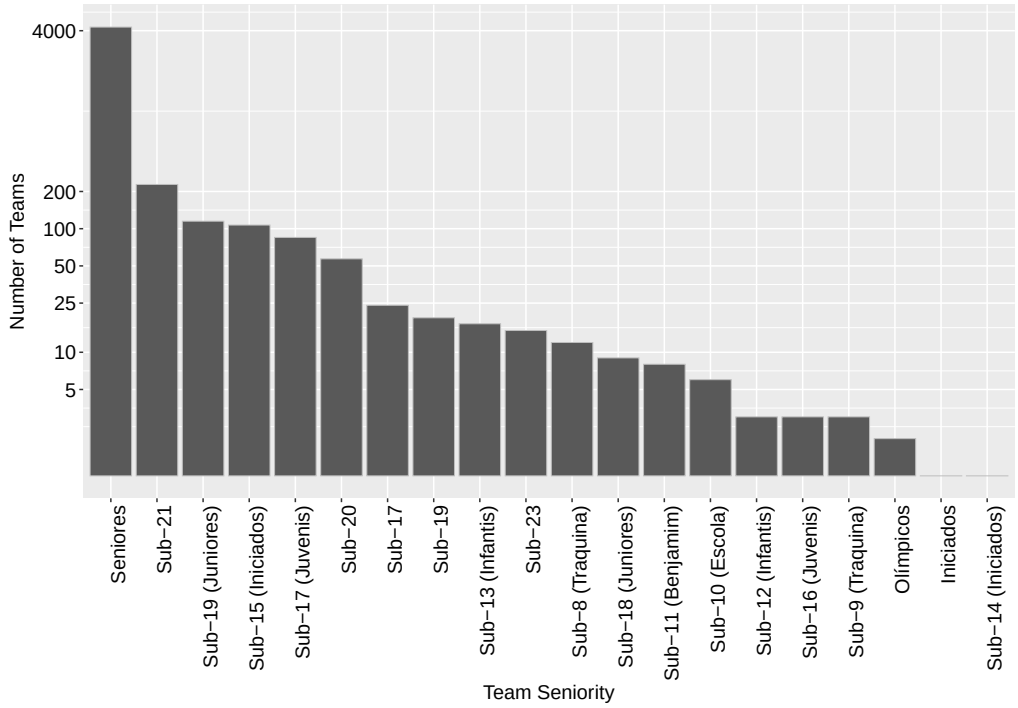


Figure 6.2: Sample Team collection distribution according to developmental stages.

6.1.3 Manager

Managers are responsible for overseeing a team in their games. There are no distinct types of manager, unlike teams and competitions seen previously.

Each one of the 3,000 sampled managers had their full name stored. Moreover, there was a keywords field that stored alternative designations, similar to nicknames, for them (e.g., José Mourinho had "Special One" stored in this field, as this is his most famous nickname). However, this field was rarely filled, with only 2.4% of sample entries containing a non-empty value associated.

Managers are people, and storing relationships between them is common. In our case, there was a field that kept relatives information about the manager in a comma-separated list of values that contain the family connection type (e.g., father, son) and the other entity's identifier. We couldn't expect this information to always be present, as footballers need not have relatives in the domain. In fact, only 13% of sample entries had this information. Ultimately, we decided to not keep it, as it could wrongfully bias search results, especially with relatives with bigger popularity differences.

Like competitions, there was a textual search field containing different expressions that users might search for when trying to find a manager. From our observation, though, these mainly seemed to consist of subsets of the manager's full name. More interesting fields included those that had information on a manager's current and past teams. These fields could help disambiguate

between managers with more common names or surnames. Finally, similarly to teams, past competitions won by a manager (with no information regarding their club at the time) helped characterize more accomplished managers. However, when competition terms are used in search, often they seek to find the competition itself and not people with past history on it, thus this field was discarded.

6.1.4 Player

Players are grouped together to form teams. Our sample contained 10,000 players. Also representing a type of person, they shared some common characteristics with the Manager entity. More specifically, the name and nickname fields were also present. However, the latter had a higher presence, with 23% of players having alternative designations. Moreover, relatives information was also kept, with a similar percentage of filled values: 12.5%.

Players could also have short biographies. These presented an overview of a player's career, along with some highlights that might not be present in other fields. However, it was only usually done only for a small subset of former well-known players, meaning that these fields had a high percentage of empty values (over 99%), rendering them not very helpful for overall search.

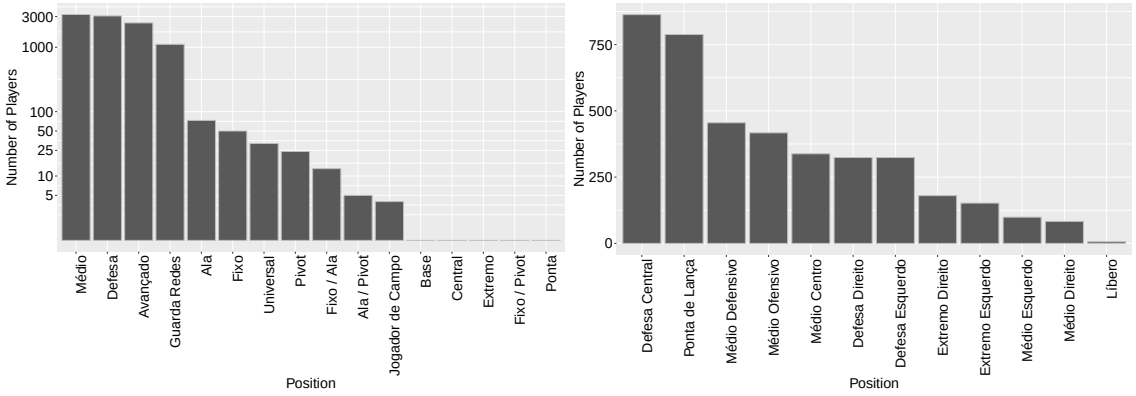
Once again, there was a textual search field that, much like in the Manager collection, seemed to represent the concatenation of a player's various designations. Furthermore, in a Player, there was information about current and past teams where he has been included, which could also help clarify what entity a user is trying to refer to when searching for players with common names. Past achievements, at individual and collective level, were also stored and helped describe a player and their career, though, once again, they present little value for searching, for the same reason identified in the Manager collection.

Finally, there were four fields that had information regarding a player's preferred positions when involved in a match. Two of them corresponded to a main and alternative generic position (i.e., goalkeeper, defender, midfield, attacker), with two more describing specific positions within each main area (e.g., a defender can prefer to play central or on one of the sides). These fields helped distinguish between players with similar, or even equal, names but that played in different infield positions, like what can be achieved through current and previous team information. Figures 6.3 and 6.4 show the distribution of values in the fields for main and alternative positions, respectively.

Midfielders represented an expected majority as it is the field area where most players lie in most team lineups. However, when going into position detail, center backs (*Defesa Central*) and strikers (*Ponta de Lança*) were most common as preferred positions.

6.2 Resource Description

The analyzed collections resulted from partial database dumps and were, therefore, very normalized, with associations between tables represented by foreign keys. Hence, to analyze them, there



(a) Player main generic position distribution.

(b) Player main sub-position distribution.

Figure 6.3: Player sample main position values distribution.

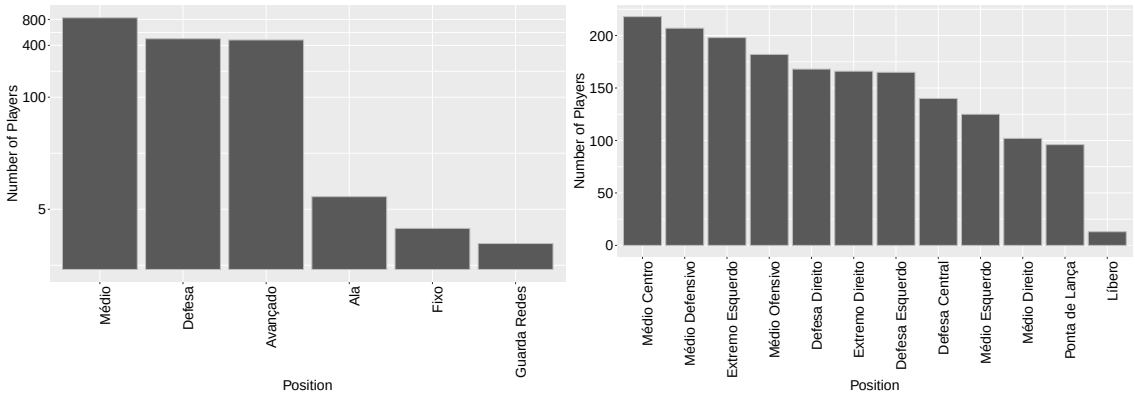
was the need to manually join the main entity dumps with other auxiliary dumps, much like INNER JOINs in SQL. When transitioning to Solr, denormalization must occur. Therefore, for each entity, the fields first filtered were reassessed in order to minimize redundancy and an unnecessarily large number of fields. Moreover, since our solution relied on Federated Search with independent engines, each entity had their separate, dedicated schema which was defined separately. Note that, being in a cooperative environment, the goal was to obtain a meaningful representation of each resource so that querying would be most effective. Furthermore, we did not consider the *Resource Selection* step: given that the number of collections we worked with was small, and that each one's size did not exceed a few thousand, performance wasn't expected to suffer significantly by always querying all collections.

In Competition entities, the only extra processing step involved checking for redundancy in common search strings stored per entity. Listing 6.1 shows the final data structure for this entity. Most values in this field corresponded to expressions already present in the name or abbreviation(s) and were, then, redundant. Consequently, we removed expressions from that field that already occurred in the other two previously mentioned.

Listing 6.1: Competition final data structure example.

```
{
  "id": "competition999999",
  "name": "Liga Espanhola",
  "abbreviation": "La Liga",
  "keywords": ["liga bbva"],
  "tier": "Seniores"
}
```

For Team entities, there wasn't as much redundancy. Listing 6.2 shows an example document with the final structure. However, some fields were unnecessarily separated and, therefore, we



(a) Player second main generic position distribution. (b) Player second main sub-position distribution.

Figure 6.4: Player sample second position values distribution.

merged them together. More specifically, old club names were split among at least three different fields, depending on the old name's origin, and were all merged into a single one. Abbreviations and initials were also joined and stored together instead of separately. While we also considered adding nicknames to the latter, ultimately they were kept separate as they represented distinct designations and not shortenings.

Listing 6.2: Team final data structure example.

```
{
  "id": "team999999",
  "name": "Impulsora Deportivo Necaxa SA CV",
  "abbreviations": ["Necaxa"],
  "keywords": [],
  "nicknames": ["Rayos","Once Hermanos","Campeoneisimo","El Equipo de la Decada"]
  "city": "Aguascalientes",
  "tier": "Seniores",
  "old_names": ["Atletico Espanol"]
}
```

Manager entities had considerable redundancy when it came to possible search strings, since, as we noticed during analysis, they consisted mostly of merges between name subsets, abbreviation and keywords. Listing 6.3 shows the final structure for an example manager. The main transformation was, then, to eliminate all this redundancy and keep only entries that actually added new vocabulary for the entity. After this process, the fields for both keywords and possible searches were merged. Despite the abbreviation field containing some redundancy, it represented names (that may be repeated or not) by which those people were more commonly referred to and, therefore, were kept in a separate field.

Listing 6.3: Manager final data structure example.

```
{
  "id": "manager999999",
  "name": "Silvino de Almeida Louro",
  "abbreviation": "Silvino Louro"
  "keywords": [],
  "current_team": "Manchester United",
  "former_teams": ["Benfica", "FC Porto", "U. Leiria", "Chelsea", "Internazionale", "Real Madrid"]
}
```

Finally, Player entities went through a similar process to managers. Listing 6.4 contains an example with this final structure. Search strings that were also mainly variants of a player's name and redundancies were mostly removed. Next, a merge between this and the keywords field occurred as well. Furthermore, abbreviations were also kept in a separate field. As players were present in bigger number when compared to managers, the distinction assumes larger importance in this entity for better search. In-field positions were mostly kept, with the exception of the alternative generic position, which was removed at this stage.

Listing 6.4: Player final data structure example.

```
{
  "id": "player999999",
  "name": "Ole Gunnar Solskjaer",
  "abbreviation": "Solskjaer"
  "keywords": ["solskaer"],
  "teams_played_in": ["Clausenengen", "Molde", "Manchester United"],
  "position": "Avancado",
  "subposition": "Ponta de Lanca",
  "alt_subposition": ""
}
```

All fields presented in all entities used the same field type, with the Solr *multivalued* property activated when fields stored arrays of values. In spite of the availability of several predefined data types by Solr, none fulfilled our requirements. More specifically, none considered the possibility of querying terms with or without accents. Therefore, we defined a custom type, *asciiFoldedName*. It used Solr's standard tokenizer, as well as an ASCII folding filter (to convert all characters to an

ASCII representation, while keeping both copies stored) and a lowercase-converting filter. Listing 6.5 shows the definition of this new field type.

Listing 6.5: Custom string field type created definition.

```
{
  "name":"asciiFoldedName",
  "class":"solr.TextField",
  "analyzer":{
      "tokenizer":{
      "class":"solr.StandardTokenizerFactory"
      },
      "filters":[
          {"class":"solr.ASCIIFoldingFilterFactory", "preserveOriginal":true},
          {"class":"solr.LowerCaseFilterFactory"}
      ]
  }
}
```

6.2.1 Search Term Payloads

In our solution proposition in Chapter 4, we aimed at using previous individual search terms to help boost document relevance for similar future searches, i.e., that used the same terms. Indeed, the schemas presented above actually incorporated one extra field, *searchTerms*, that stored these terms, alongside a search relevance weight for each one. We used the basis provided by Oakes et al. [59] to calculate search term relevance. However, this relevance was used as complementary to the overall document relevance score calculated by the search engine, instead of a replacement (as done by them). The difference also shows how the goals between both works differed: in the original proposal, authors aimed at exploring how this technique could aid in multilingual search, while we intended to facilitate previous search pattern recognition for future searchers. More specifically, given an entity e, for each query term t, they defined its search relevance with a TF-IDF like weight as:

$$Weight(t,e) = TF_{t,e} \times \log \frac{N}{E_t} \tag{6.1}$$

where $TF_{t,e}$ is the number of searches for entity e that use term t, E_t is the number of entities whose term t led to a click in it and N is the total number of documents (here, limited to our samples' size).

This weighting scheme, despite being based on a traditional relevance formula, allowed to highlight terms that were often used to reach an entity, were seldom used to search others, or even ideally, both. In order to verify an optimal weighting scheme, and because term frequency differences between distinct terms for a given entity were sometimes considerable, we implemented two variants. The first is as described, while the second utilizes logarithmic term frequency (see Figure 3.4).

To incorporate this into collection documents, we made use of a Lucene feature that was recently ported to Solr: payloads [36]. The idea behind this feature is to associate some score to individual terms. Use cases vary, and what suited ours was that it helped weight terms in the form of relevance confidence in it.

Once again, while Solr provided predefined field types, they didn't cover our full necessities, hence we defined our own payload field type that used floating-point weights. Listing 6.6 shows the corresponding definition and an example below it. The difference mainly resided in the application of the ASCII folding filter, fulfilling the same necessity as our other custom field type. Notice that the tokenizer here only splits on whitespace, as it is how payloads are delimited by implementation design.

Listing 6.6: Custom payload field type created definition and example.

```
{
  "name":"searchTermsPayload",
  "class":"solr.TextField",
  "analyzer":{
       "tokenizer":{
          "class":"solr.WhitespaceTokenizerFactory"
       },
       "filters":[
            {"class":"solr.ASCIIFoldingFilterFactory", "preserveOriginal":true},
            {"class":"solr.LowerCaseFilterFactory"},
            {"class":"solr.DelimitedPayloadTokenFilterFactory", "encoder":"float"}
       ]
    }
}

//Payload example
{
  "searchTermsPayload": "term1|50.0 term2|25.5 term3|1.0"
}
```

6.3 Architecture

Despite the actual search process responsibility being delegated to Solr, we were not able to implement a fully federated solution within it. The tool was mostly prepared to perform Distributed Search, i.e., distributing some collection's documents over several, independent shards. However, what we needed was a way to simultaneously query multiple collections and normalize the respective resulting documents' scores according to some provided formula. Therefore, there was the need to introduce another layer, an extra server, that interacted with Solr to query collections and then performed merging locally. We opted to use Python's Flask [65] due to being lightweight and easy to setup and use. Figure 6.5 illustrates the adopted architecture in a simple diagram that also highlights the main execution flow steps.

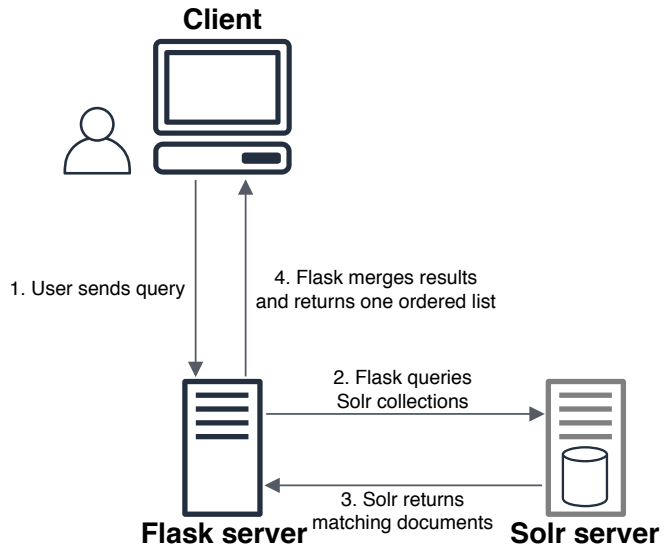


Figure 6.5: Overall system architecture and main execution flow steps.

When a user submitted a query, the server would build the request object that was sent through the Solr API, for each one of the collections. Upon receiving all results, score normalization was performed, before sending a final ordered list back to the client. Because we were working with database samples that were processed to only include relevant, searchable information, and not the production system itself, there was no external database connection. In a realistic deployment scenario, the Solr server could connect to the database in order to query all necessary information for presenting results, though the usage of stored, non-indexed fields in Solr is equally valid and used in practice. Moreover, for simplicity, there was a single Solr instance running, with all collections residing in it.

6.4 Execution Flow

The steps presented in the architecture diagram provide an overview of how the system operated when a query arrives. However, in each step, there were considerations that had to be taken into account. For example, when obtaining results from all collections, each had to be queried independently, using Solr's API. The request wouldn't be the same for every collection: search fields and respective boosts were unique to each one.

Moreover, there was no direct way to dynamically incorporate search term payloads into the total document score, especially if we wanted to configure multiple ways of integration. Finally, when merging results, we had to normalize all document scores by taking into account each collection's expected quality.

In the next subsections, we discuss how we implemented each of these tasks.

6.4.1 Querying Independent Collections

During step 2 shown in Figure 6.5, the Flask server propagates the query to each collection. However, given the difference in schema, requests had to be adjusted according to the entity that was being queried.

We started by constructing the common core of the request. This included defining the query parser and general options. We opted to use Solr's eDismax query parser [71] due to its range of capabilities and easiness to modify, e.g., field-dependent boosts. In addition, the *stopwords* parameter was set to false, so that these terms weren't removed during the analyzer pipeline. Moreover, despite the existence of a dedicated Payload query parser, that wasn't what met our needs: we intended on calculating document relevance by Solr standards *and* by term payloads, not one or the other. Therefore, we used eDismax to calculate traditional document relevance and made use of the in-built *payload* function and possibility of defining custom, calculated fields for each returned document. This function takes a payload field and a term, returning the respective weight defined, or 0 otherwise. Upon receiving a query, the server split it using a whitespace delimiter and created one custom field per term. Each term was assigned a sequential field (payload_0, payload_1, ...) that searched the *searchTerms* field common to all entities. Listing 6.7 shows this process, while Figure 6.6 illustrates a small example of how the server obtained search term payload values for later incorporation.

Listing 6.7: Custom term payload fields making for a query request.

```
# Split query into its terms
query_terms = [term.strip() for term in query.split(' ') if len(term) > 0]

request_params = dict()
...
# Build custom fields for payload scores for each query term
```

```
request_params['fl'] = '*,score,' + ','.join(['payload_{}:payload(searchTerms,{})'.format(idx, term) for
    idx, term in enumerate(query_terms)])
```

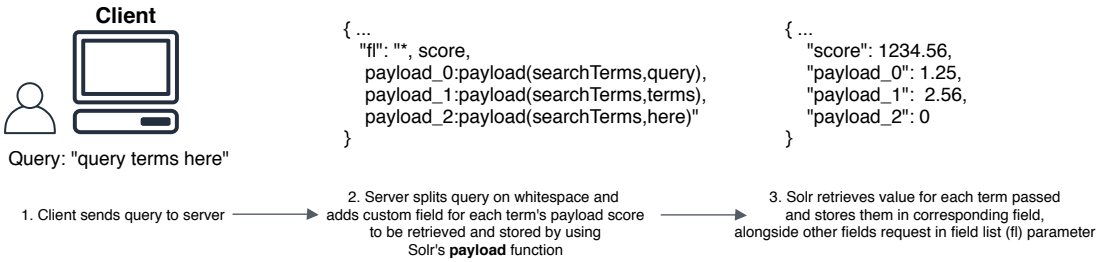


Figure 6.6: Example of how server obtains search term payload values. Notice the definition of custom fields through the syntax fieldName:<*calculation*> and respective inclusion in the field list (*fl*) on step 2. After step 3, the server can filter fields that start with *payload_* to obtain all values for payload integration.

Given this common core, we added the search fields, along with respective boosts, for each entity, so that we could take advantage of the data structure to value matches on more relevant fields. Furthermore, we replicated these boosts into a phrase search parameter. This way, not only could we boost documents that matched terms in a given field, but also on consecutive term occurrences, much like phrase querying. Table 6.1 shows the field boosting definition for each entity type. The weights presented were a result of *a priori* experimentation. Moreover, we limited phrase query match to a slop of 3 terms (*ps* parameter), as well as configured Solr to retrieve all matching documents (by default, only the top 10 are included).

Table 6.1: Field boost definition per entity type.

Entity	**Field boosts**
Competition	name^10 abbreviations^10 keywords^5 tier^5 searchTerms
Player	name^10 abbreviations^7.5 keywords^7.5 teams_played_in^5 position^5 sub-position^5 alt_subposition^3 searchTerms
Manager	name^10 abbreviation^7.5 keywords^7.5 current_team^5 former_teams^3 searchTerms
Team	name^10 abbreviations^10 keywords^7.5 nicknames^10 city^5 tier^3 old_names^3 searchTerms

6.4.2 Incorporating Payload Scores

Having queried a collection, the resulting documents all had a *score* field, with the estimated Solr relevance formula result, as well as *payload_** fields, one for each query term issued. To merge these scores, we used a set of strategies based both on linear scaling and linear combination of

both factors and terms, respectively, so that we could later assess the most effective strategy. To this end, given a document relevance score *RelS* and payload scores $PL_{0..n}$ for that document, we defined and experimented with 4 strategies.

In the first strategy, *Prod*, the document's final score linearly scaled with the matching payload scores, that is:

$$Score = RelS \times \prod_{i=0}^{n} PL_i[PL_i \neq 0]$$

On the other hand, the *Sum* strategy performed a linear combination between the relevance score and the sum of the matching payload scores:

$$Score = \alpha \times RelS + (1-\alpha) \times \sum_{i=0}^{n} PL_i, \alpha \in [0,1]$$

where the α parameter could attribute more or less importance to term payloads and defaulted to 0.5.

The *ProdLog* strategy is similar to *Prod*, but the sequence product of the payloads was log-smoothed, i.e.,:

$$Score = RelS \times \log \prod_{i=0}^{n} PL_i[PL_i \neq 0]$$

where the log value should evaluate to a positive number, otherwise it wouldn't be considered.

Finally, the *SumLog* incorporation strategy was derived from *Sum*, similarly to *Prod* and *ProdLog*:

$$Score = \alpha \times RelS + (1-\alpha) \times \log \sum_{i=0}^{n} PL_i, \alpha \in [0,1]$$

where the payload component, unlike *ProdLog*, was always considered, since there was no risk of nullifying the final document score (unless $\alpha = 0$).

6.4.3 Results Merging

The last step in the retrieval process was the merging of results from all four collections in order to produce one ordered list. To merge documents, we had to normalize their scores, so that they could be comparable between different collections. One of the most well-known algorithms for this task is CORI [19]. For a given document D retrieved from collection C, CORI defines its normalized score as:

$$FinalScore = S_D * \frac{1 + 0.4 * S_C}{1.4}$$

where S_D is the original document score (post term payload incorporation) and S_C is the collection's score, while the normalization constants are a product of experimentation.

A collection's score should reflect their retrieved documents' overall importance for the final ordered list, thus its value could be the difference between a document placing in the top 10 or much further down. We followed the work of Hawking et al. [52], that used a LMS strategy to calculate a collection's score. LMS, short for "using result Length to calculate Merging Score" [63], requires no collection metadata or samples, ranking collections based on the result set size for a given query. More specifically, a collection's LMS score is defined as:

$$LMS_C = \log\left(1 + \frac{|R_C| \times K}{\sum_{i=1}^{n} |R_i|}\right)$$

where $|R_i|$ is the size of the result set returned by collection i for the query, n is the number of collections to merge, and K is a scaling constant. In the original proposal, the authors set a value of $K = 600$. This is also the value we adopted.

The strategy's simplicity came with the trade-off that it might overvalue a collection's worth just for returning a larger set of documents, even if most have little to no relevance to the underlying information need, or even the converse (e.g., a collection that returns only a few documents, but one of those is the most relevant). In our case, term overlap between collections wasn't expected to be as common (except in the case of managers and players), so a larger result set could help highlight, at least, the correct entity type the user was looking for. Moreover, when combined with effective document scoring techniques, it has been proved to produce good results: both Rasolofo et al. [63], in their original LMS proposal, and Hawking et al. [52] have come to this conclusion, with the latter concluding that the strategy had no significant differences from other, more complex ones. Here, search term payload incorporation could help mitigate LMS' effect in case of a wrongful judgement of a potentially relevant document.

6.5 Summary

Before implementing our Federated Search solution, we briefly looked at the target collections and analyzed the distribution of some main fields in each one, finding common football patterns from the samples. We then projected a prototype with an Apache Solr server that stored one separate index for each entity type, and a Python Flask server to normalize and merge results across collections. To store previous search terms, Solr's payloads feature was used, which allow individual term weighing. Each term's weight is derived from a TF-IDF adaption, reflecting how often it was used to reach that and other documents. To incorporate payload values, we defined four strategies as a product of two binary variables: score update (linear scaling or linear combination) and previous search term frequency (raw value or log-smoothed). Finally, for merging and normalization, we made use of previous work on CORI variations that used returned result set size as a main signal for collection quality.

Chapter 7

Solution Evaluation

With a defined solution, we subjected it to an evaluation process in order to assess its quality and capability of fulfilling the intended purpose. For this, we used different sets of queries, with different characteristics, and evaluated their results under several metrics so as to avoid potential bias of a reduced number of them.

In order to annotate test queries with the correct answers, we used click information as a relevance proxy. In Chapter 3, we pointed to the work of Joachims et al. [44] that exposed the potential dangers of using click-through information to this end. While we also argued that, due to the context of our engine, where little free text was shown and, therefore, curiosity or further confirmation clicks were expected to be rarer, it is important to note that seldom can clicks actually replace manual human judgement. This was another reason why a diverse set of metrics was adopted. In order to validate obtained results, behavior across metrics should remain reasonably consistent. Finally, when possible, we cited work that supported the usage of click-through data as an acceptable replacement for manual relevance judgement.

We then start by describing what were the query sets used and what distinguished them. Next, we describe the evaluation metrics used, i.e., their origin and intent. Finally, we present the obtained results, setting the current production engine at zerozero.pt as our baseline, and discuss our findings.

7.1 Datasets

To assess the developed system's quality, we used two distinct sets of queries, so that we could analyze the engine's behavior under different circumstances and scenarios. The first set of queries contained popular queries in terms of frequency. We have already noted that the current production engine provides satisfactory result sets for queries that were common. Therefore, it was of interest to assess if our solution did not bring an associated cost of reduced capability of answering these types of queries. To build it, we collected the 200 most frequently submitted queries to the system from the analysis log and all the entities clicked as a result of that search, keeping only the most clicked, alongside the number of clicks it received. In their work, Liu et al. [53] concluded that,

for navigational queries, it was possible to automatically annotate queries with the most clicked result as the correct answer, obtaining over 96% accuracy on the annotation process when compared to manual judgements. Indeed, in our context, queries were expected to be predominantly navigational, as users mostly sought a specific entity when searching. Moreover, on average, the top clicked result in each of the 200 queries had an average click share of 85%. Figure 7.1 shows the full distribution of click share for top clicked results.

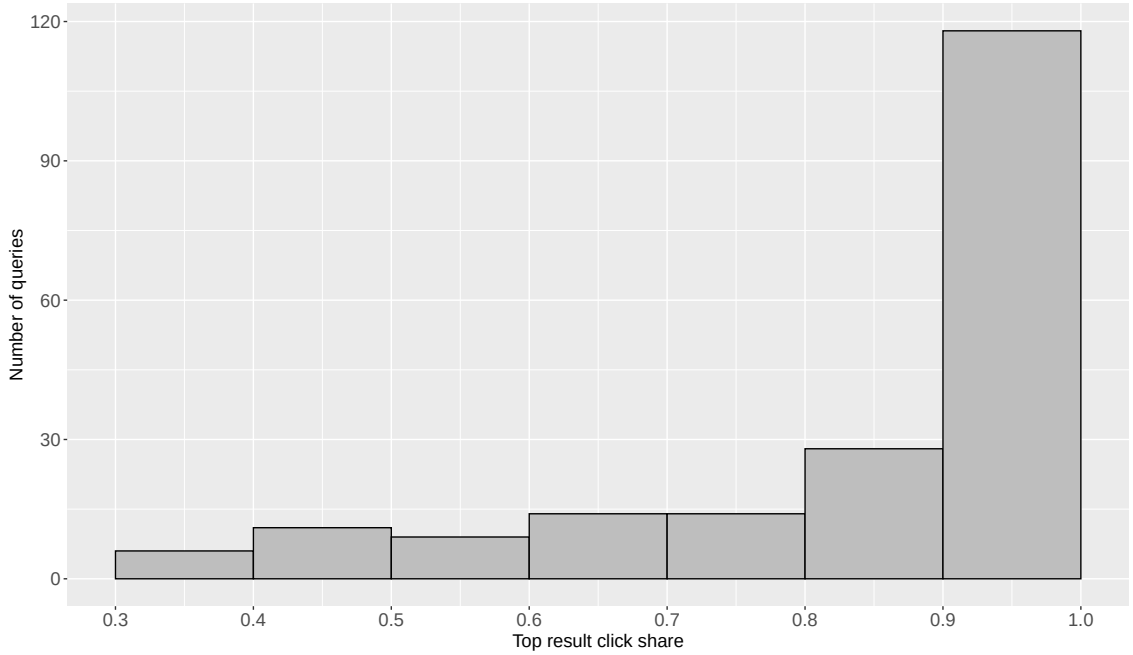


Figure 7.1: Click share distribution for top clicked result (Frequent Query Set).

We could see that, for nearly 120 queries of the set, the top clicked result had collected over 90% of all clicks for that query. Such skewness towards one result was also a good indicator that helped validate the automation of the annotation process. The final step in the building process was to remove queries whose correct answer was not available in our collection samples — 16 queries were removed, leaving us with 184 queries to constitute this evaluation set.

The second set of queries consisted of interrogations that produced higher variability in clicked results, i.e., different entities were considered the correct answer depending on the searcher, and none of them had a noticeable click share majority, unlike the previous query set. This corresponded to the top queries with most entropy, as measured during our analysis in Chapter 5 (see Tables 5.18a and 5.18b). In this case, we filtered queries based on individual entity click entropy. The building process followed a similar flow to the frequent queries set. We started by collecting the 200 queries with highest entropy value. However, this time, we considered all results clicked as potential answers. With an average click share of 36.9%, it was harder to assume that the top clicked results were the only correct answer. Figure 7.2 shows the full top result click share distribution for this query set.

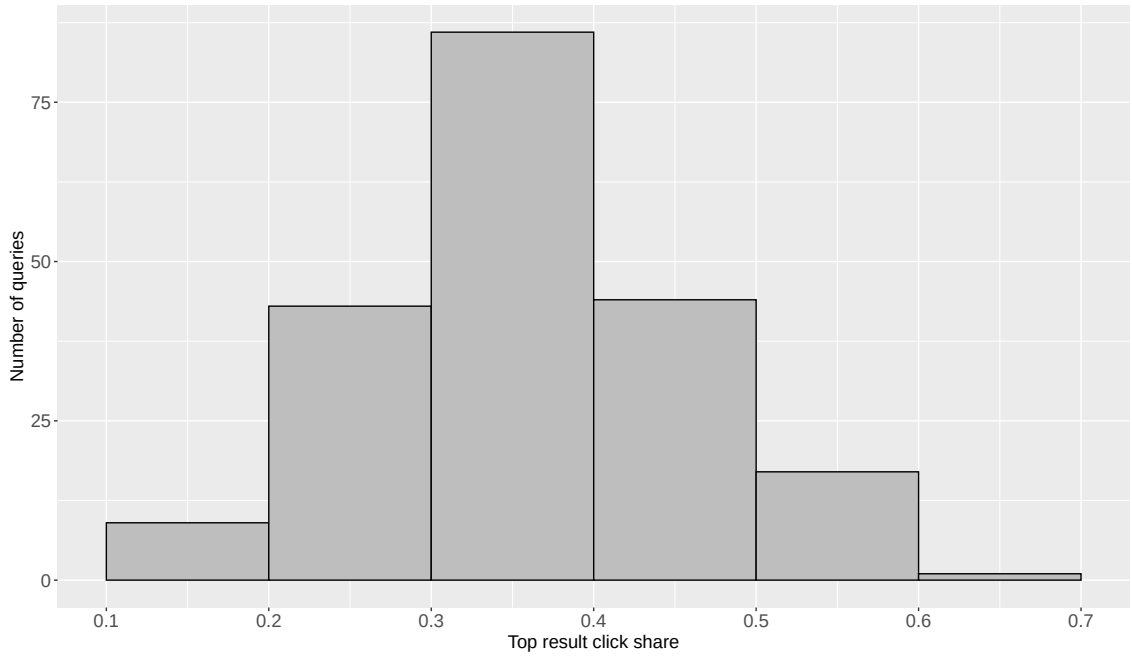


Figure 7.2: Click share distribution for top clicked result (Entropy Query Set).

A pattern more resemblant of a normal curve was observed. Here, the most common click share for the top result was between 30% and 40%, with well over 75 queries falling into that range. Moreover, due to the ambiguous nature of these queries, it was necessary to consider multiple possible answers. Once again, all answers had attached the number of clicks received and we excluded queries that contained no possible answers present in our collection samples. This left us with 179 queries for this set. With it, we intended to validate the hypothesis that our solution could improve upon the current situation, as higher entropy queries weren't usually as frequent and, therefore, not tuned to the current production engine.

Accordingly, these query sets will be referred to as *Frequent Query Set (FQS)* and *Entropy Query Set (EQS)* from now on.

7.2 Evaluation Metrics

To evaluate the robustness of the system, both query sets were tested against different metrics. As we had no access to a test collection with full relevance judgements for all documents, we avoided measures that directly dealt with recall. Note that most metrics were system-wide, sometimes an aggregate of local query level metrics, to more easily compare systems. In addition, all metrics were used in testing both query sets, unless stated otherwise.

The *Mean Reciprocal Rank (MRR)* [85] is the arithmetic average of every query's *Reciprocal Rank (RR)*. For a given query, the reciprocal rank is defined as the inverse of the ranking of the highest-scored correct answer, i.e., if the highest ranking for a correct answer was r, then their RR

would be $\frac{1}{r}$. Due to only considering one document, it is often most used and effective when each query has only one correct answer, i.e., navigational queries, such as ours.

Despite being, in practice, a weighted click-through metric, it doesn't incorporate click-through frequency. Walter Underwood [83], an experienced researcher with tenures dedicated to search engine development and optimization at Netflix and Ultraseek, amongst others, decided to expand traditional MRR and add click-through frequency information, so that each RR is weighted by a query's click frequency. In other words, a query's *weighted Reciprocal Rank (wRR)* would be now calculated as follows:

$$wRR(q,r) = RR(r) \times ClickFrequency(q)$$

A system's *weighted MRR (wMRR)* would then be, again, an average across queries, this time weighted (i.e., the denominator is now the total number of clicks). This way, the metric values more frequently chosen results at higher positions.

A metric that is commonly used is the *Mean Average Precision (MAP)*. Similar to MRR, it is the result of the average of another metric applied at query level, this time *Average Precision (AP)* [16], a metric regarded as very stable. For a given query, its AP is defined as the average of the precision values obtained for the set of top N documents existing each time a new relevant document is retrieved. As an example, for a given query whose correct results were returned on the 1st, 3rd and 10th positions, its AP would be $\frac{1/1+2/3+3/10}{3} \approx 0.66$. Note that, in the case of a single correct result, both AP and RR (and, therefore, MAP and MRR) are necessarily equal. There is also a variant that performs a geometric average instead (GMAP). However, we only used MAP in our evaluation.

While studying how to achieve better precision for Intranet search systems, Zhu et al. [101] presented an evaluation criteria that assessed the presence of correct answers in the top results as a cover measure. They defined *Success (at N)* as the percentage of queries that had at least one correct answer in the top N results in the result set. Note that this metric, unlike previous ones, isn't the result of an aggregation of query level metrics. With it, we could more easily assess the placement behavior of each system regarding the answers annotated as correct. In their study, Zhu et al. measured this metric for values $N = \{1,5\}$. We replicated this choice.

All previous metrics were executed against both query sets. However, for the Entropy Query Set, we performed two further measurements that assessed the similarity between the ranking produced by an engine and a hypothetical ideal ranking based on click frequency. The first measure in this regard was the *Normalized Discounted Cumulative Gain (nDCG)* [43]. This metric is useful when relevance isn't binary, but instead lies on a graded scale and contemplates multiple answers. It is based on the more primitive *Cumulative Gain (CG)* that corresponds to the sum of graded relevance for all results returned. The difference in the discounted version is the penalty factor for relevant documents being positioned in lower rankings. Often, the metric is applied to results up until a rank threshold r. Its DCG can be defined as:

$$DCG(r) = \sum_{i=1}^{r} \frac{rel_i}{\log_2(i+1)}$$

where rel_i is the graded relevance score for the document returned at rank i (0 if not relevant).

Since results may present highly variable values between queries, DCG value ranges aren't easily comparable. In order to do this, one can sort all relevant documents by their graded relevance score and calculate what would be an ideal DCG value for that ranking (IDCG). Of course, this relies on having complete relevance information. Here, click frequencies corresponded to an approximation for this graded relevance. The normalized score, $NDCG(r) = \frac{DCG(r)}{IDCG(r)}$, is then comparable and can be, for example, averaged to produce a final system score.

Finally, and also only on the Entropy Query Set, we calculated, for each query, its Spearman's rank correlation coefficient [72]. This coefficient assesses if the relationship between two variables from dependent samples can be characterized as a monotonic function, i.e., both variables grow in the same direction. In our case, we measured the similarity between the ranking positions of relevant documents returned and the ideal ranking of these documents. For example, if relevant documents, ordered by relevance, were returned in positions 5, 3 and 10, then the coefficient would analyze the similarity between variables $X = [5,3,10]$ and $Y = [1,2,3]$. However, unlike other metrics, we didn't directly aggregate coefficient values. Instead, we performed statistical hypothesis testing to assess if coefficient values varied significantly between different strategies in our solution and, most importantly, if there were improvements over the baseline.

7.3 Results and Discussion

We now present the results obtained, partitioned by query set. All metrics were applied with a threshold at ranking 10, i.e., their values were calculated considering only the top 10 results. The only exceptions to this were, naturally, the Success@N metric variants used.

In the following subsections, we considered each independent combination of search term payload score calculation and incorporation as an independent search engine. Each combination followed a similar naming convention: *(P|L)STRATEGY*, where P is for PureTF, L for LogTF and STRATEGY is one of the incorporation mechanisms described in Section 6.4.2. For example, the engine resulting from the combination of Pure TF (in payload calculation) and the product operator (in payload incorporation) would be referred to as *PProd*. *zerozero* will represent the baseline that is the current product engine. Finally, *NoPayload* would refer to a local baseline for our solution, where there was no search term payload usage, i.e., *plain federated Solr*.

Regarding Sum and Sumlog strategy variants, we experimented, *a priori*, with α values in the range [0, 1], with a step of 0.1, in order to assess the best linear combination weights. Optimality was achieved when $\alpha = 0.3$, that is, the document's score was made mostly from search term payloads (70%). As we approached both extremes of the tested range, performance tended to get worse, as expected. However, this effect was most felt when raising α, meaning that Solr's relevance score alone wasn't enough to produce satisfactory results.

7.3.1 Frequent Query Set

Results obtained for the Frequent Query Set are shown in Table 7.1 and in a multi-bar plot for easier comparison in Figure 7.3. For this query set, the zerozero baseline was generally the highest performing strategy, hence it was isolated in the table. Moreover, we highlighted, for each metric, the strategies that had the highest value.

An immediate conclusion was the performance of the local baseline, *NoPayload*, ranking worst in every metric calculated. Particularly, in only around 47% of the queries was the engine able to return the correct result in the first position, and only in a little over two-thirds placed it in the top 5. This was further corroborated by the lower values of weighted and unweighted MRR, reflecting the typically lower ranking positions where correct results were returned. Note that, by considering only one correct answer per query, unweighted MRR and MAP values were equal, therefore we showed only one designation, in this case the former.

Regarding the different combinations of our solution, it appeared that using Sumlog for search term payload incorporation, regardless of TF type used for its calculation, lead to worse results. In fact, when using TF Log, results were almost comparable to not using payloads at all. This was likely a sign that both the Sumlog incorporation strategy and logarithmic TF calculation flattened payload values beyond significance. The latter's effect could be seen on the other variants as well: when compared to their Pure TF counterpart, no variant performed better on any metric. In the remaining three strategies (*PProd*, *PSum* and *PProdlog*), performances were more similar and close to the zerozero baseline. As for the other two, they were almost indistinguishable performance-wise. When compared to the zerozero baseline, results were also very similar, with a small advantage for the baseline in the Success@5 metric. In that case, the current production engine had slightly better performance, with the 2 percentage point difference corresponding to only around 4 queries. Indeed, frequently searched entities already produced good results, as was shown by these results. Therefore, the main conclusion was that some of the combinations of our solution were capable of replicating the good results already provided by the baseline for these situations.

Finally, we focused on one metric and looked at its individual values for all queries across one of the top combinations, *PProd*, and the zerozero baseline. This way, we could assess if results weren't biased due to some outliers. We chose Average Precision (AP) as the metric to observe, not only due to its robustness, but also since it considered multiple answers per query, an important factor so that we could more confidently perform the same analysis later on the Entropy Query Set. In the Frequent Query Set, this also corresponded to the MRR, due to the one correct answer only assumption. Figure 7.4 shows the obtained values.

As expected, values were very close, reflecting what was obtained in their averages. Despite being able to keep a better performance for a short while, the current production engine then had a slower rate of descent for the lowest scoring queries.

Table 7.1: Evaluation metrics results (Frequent Query Set).

Strategy	MRR@10	wMRR@10	Success@1	Success@5
zerozero	0.84535	0.99580	0.83696	0.87370
NoPayload	0.56864	0.66987	0.47283	0.68478
PProd	**0.84375**	**0.99508**	**0.83696**	**0.85326**
PSum	0.84013	0.99476	0.83152	**0.85326**
PProdlog	0.82201	0.97689	0.79348	**0.85326**
PSumlog	0.66498	0.78311	0.56522	0.78804
LProd	0.80471	0.95340	0.76630	**0.85326**
LSum	0.68554	0.83061	0.59239	0.80435
LProdlog	0.76771	0.91409	0.70109	0.84783
LSumlog	0.60714	0.70529	0.51630	0.73370

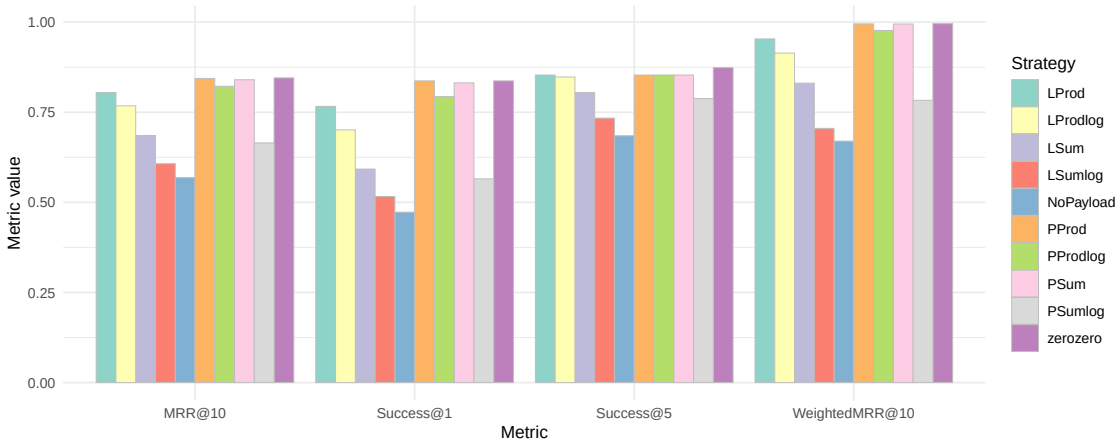


Figure 7.3: Evaluation metrics results (Frequent Query Set).

7.3.2 Entropy Query Set

Evaluation results for this query set are presented in Table 7.2 and Figure 7.5. Once again, for each metric, the best strategies' values were bolded. Furthermore, note that, while the NDCG column label reads *NDCG@10*, the values presented are the arithmetic average for all queries, and not individual values.

As expected, some metrics had lower values when compared to the Frequent Query set, since it was harder to correctly serve this kind of queries. Once again, the local baseline of not having search payload incorporation achieved the worst performance of any combination, a repeated behavior. Only close to 40% of queries had a relevant result returned first in the ranking, with the MRR value revealing that, on average, documents tended to place second. The MAP, which was equivalent to MRR in the Frequent Query Set, here also followed a similar pattern to it, though with even lower values. Finally, the new metric, NDCG@10, also suggested that there wasn't

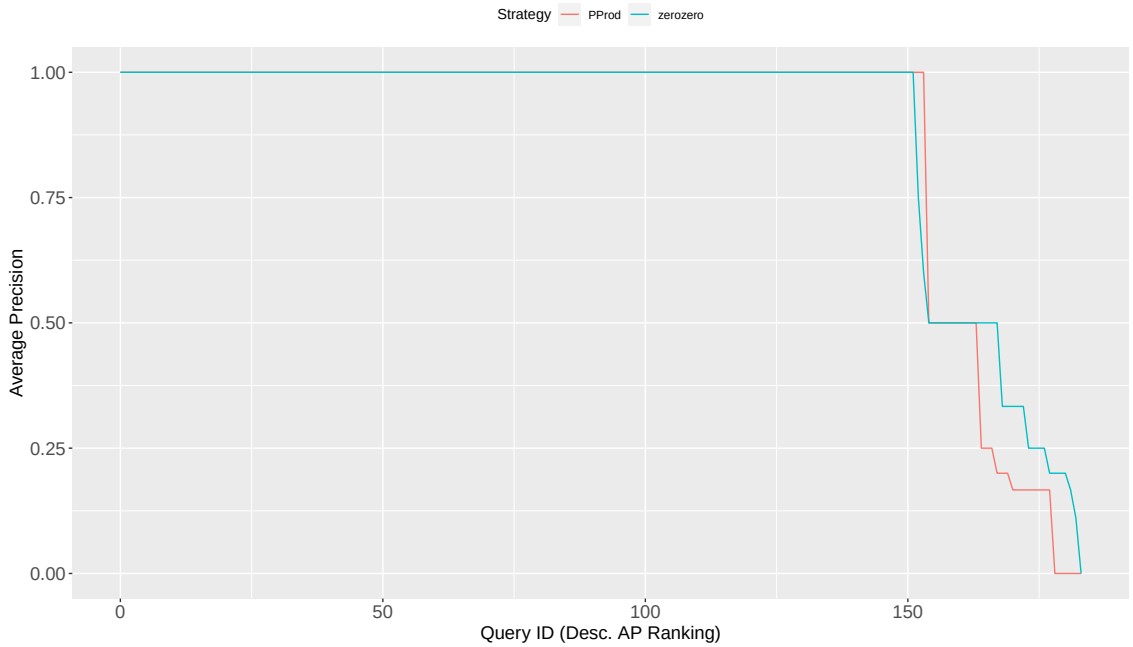


Figure 7.4: AP values per query in descendent order (Frequent Query Set).

much gain as we moved in the ranking when compared to other strategies. Being a pattern present in both query sets, this then confirmed that payloads were necessary for improving retrieval performance in our solution.

Sumlog strategy variants were, once again, the ones with lowest performance, demonstrating the low expressiveness of payloads in that scenario. However, contrary to the Frequent Query Set, there were more strategies that suggested an improvement over the zerozero baseline. More specifically, both *PProd* and *LProd* surpassed other strategies on all metrics (e.g., they were the only ones with a MAP value of over 0.8). Moreover, they also presented the highest discounted cumulative gain, with NDCG@10 values of around 0.82 each. In terms of correct result presence in the top 1 and 5 ranking positions, other strategies, such as *PSum* and *PProdlog*, performed nearly as well as the former two. In fact, their difference in other metrics was usually small (around three percentage points maximum, when observing MAP). These results showed how improvements were necessary for this kind of queries in the current production engine.

The last metric we used, the Spearman ranking correlation coefficient, was then tested such that we could assess which, if any, of our combinations produced rankings that were, on average, closer to the results users tend to access. We started by observing the coefficients' distribution for each one of the strategies. This is shown in Figure 7.6.

Results corroborated some of the observations made previously, e.g., no payload incorporation was the worst solution, with Sumlog strategies following. Moreover, the *PProd* strategy's values appeared to be the highest, despite having some outliers with lower correlations. While the boxplot provided visual feedback on the expected results, further objective measurements were needed, as

Table 7.2: Evaluation metrics results (Entropy Query Set).

Strategy	**MRR@10**	**wMRR@10**	**MAP@10**	**Success@1**	**Success@5**	**NDCG@10**
zerozero	0.7682	0.9540	0.7492	0.7324	0.8388	0.7451
NoPayload	0.5082	0.7187	0.4822	0.3966	0.6760	0.5198
PProd	0.8304	**0.9955**	0.8011	**0.7989**	0.8659	**0.8201**
PSum	0.8216	**0.9953**	0.7797	0.7933	0.8547	0.8035
PProdlog	0.8233	0.9927	0.7880	0.7933	0.8603	0.8038
PSumlog	0.6974	0.9133	0.6617	0.5978	0.8436	0.6979
LProd	**0.8317**	0.9932	**0.8102**	**0.7989**	**0.8715**	0.8161
LSum	0.7832	0.9683	0.7472	0.7263	0.8547	0.7648
LProdlog	0.8042	0.9865	0.7760	0.7654	0.8547	0.7825
LSumlog	0.6057	0.7458	0.5803	0.4916	0.7709	0.6043

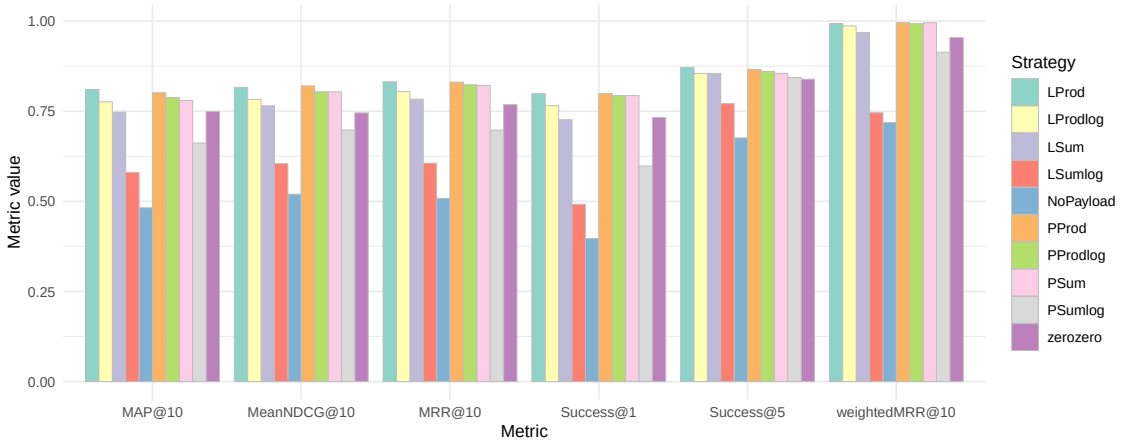


Figure 7.5: Evaluation metrics results (Entropy Query Set).

there was no standard practice for aggregation of Spearman coefficient values that would allow a more direct comparison. Therefore, we performed hypothesis tests by comparing each combination from our solution with the zerozero baseline and verifying if we could confidently state that correlation values tended to be higher.

We started by assessing the coefficient data's normality using the Shapiro-Wilk normality test [68]. The null hypothesis $H0$ states that the data follows a normal distribution, and can be rejected if the *p-value* falls under the chosen alpha, which we adopted to a standard 0.05. After running on all coefficient datasets (from our solution and the zerozero baseline), the *p-value* was always much lower than the threshold set, therefore the normality assumption failed every time. Despite the central limit theorem stating that, given a large enough population size, independently of its distribution, a sample will always tend to follow a normal distribution, we opted to discard parametric tests that relied on the normality assumption.

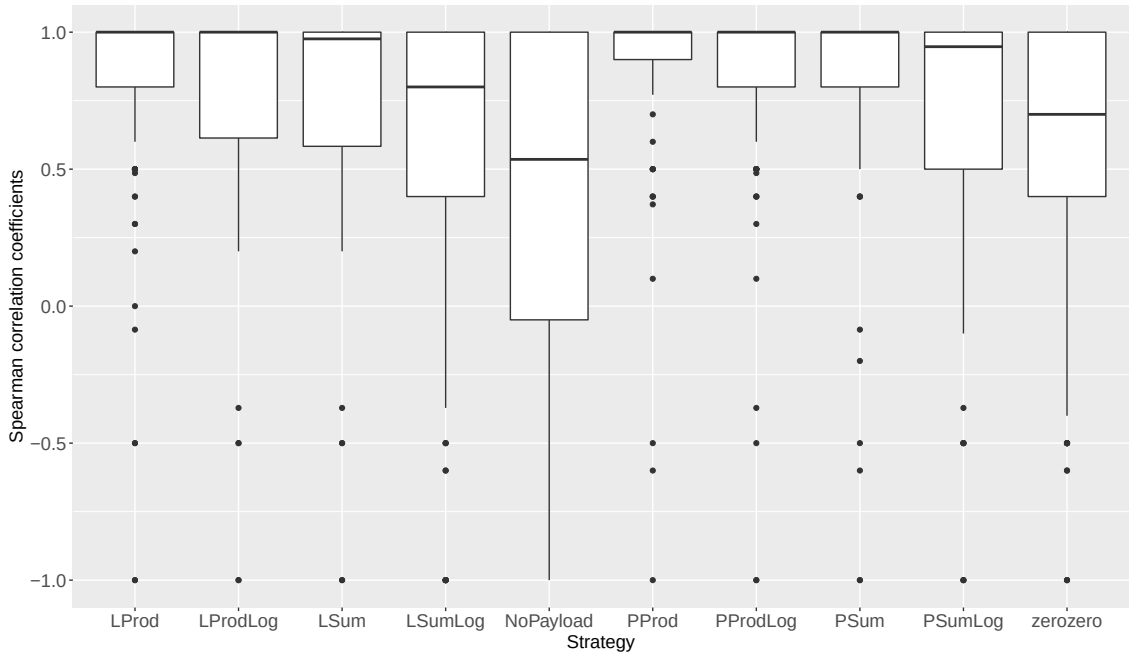


Figure 7.6: Spearman rank correlation coefficients distribution.

Our choice was then the non-parametric Wilcoxon signed-rank test [93]. This test is a reliable alternative when there is no support for data normality and can be used to assess if there are significant changes in the distribution of two variables, and if these changes are one or two-sided. Given two variables X and Y, the null hypothesis is $H0 : X \leq Y$. Our interest was then in checking which combinations were able to reject the null hypothesis. Once again, we considered a threshold of $p = 0.05$. Table 7.3 shows the results obtained. Results below the defined threshold are marked with an asterisk (*). Values lower than 0.001 are highlighted with a double asterisk (**).

Table 7.3: Wilcoxon signed-rank test *p-values*.

Strategy	***p-value***
NoPayload	0.922
PProd	1.352e-12**
PSum	1.521e-6**
PProdlog	4.444e-9**
PSumlog	0.036*
LProd	1.311e-7**
LSum	0.007*
LProdlog	3.685e-6**
LSumlog	0.517

Results showed that, for both *NoPayload* and *LSumlog* strategies, we couldn't state that the correlation coefficients were higher than the baseline. Moreover, the *PSumlog* strategy, despite

being able to reject the null hypothesis, resulted in a much higher value, attributing less confidence to this strategy as well. Otherwise, all strategies' results allowed us to say that they produced rankings with higher correlation to typical user result access patterns.

Finally, we took a look at individual AP values for this query set, much like we did for the previous one. Figure 7.7 shows the corresponding plot.

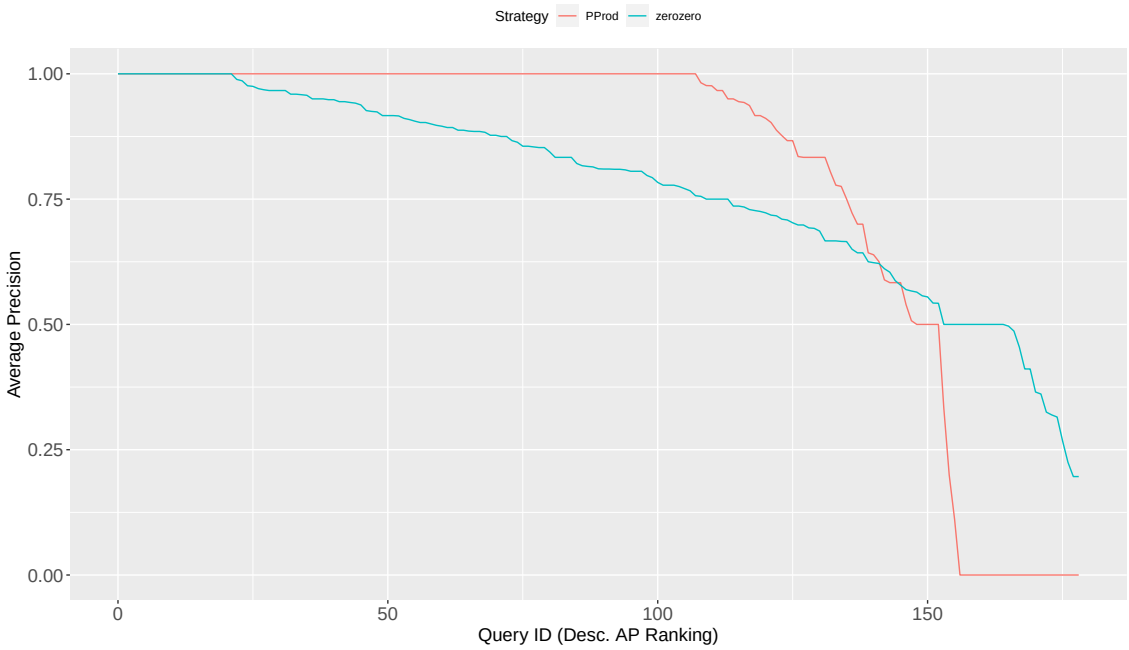


Figure 7.7: AP values per query in descendent order (Entropy Query Set).

Differences were more noticeable, also reflecting the enhancement visible when evaluating this query set. There was a sudden break for our strategy towards the worst performing queries. This was likely due to the correct answer inclusion strategy: we considered all queries that had at least one answer present in the samples, no matter how often it was clicked. For queries that closely resembled this edge case, it was not possible for our solution to place them in the top 10 answers. Overall, our solution appeared to produce much better results according to user access patterns.

As a complement, we looked at the top queries with highest difference in performance between both systems. Tables 7.4a and 7.4b show these entries. Note that the difference was between our strategy and their baseline. Therefore, a positive value indicated an improvement, while a negative value dictated the opposite.

Improvements were more significant than the worsening effect for the lesser performant queries. Amongst the queries with highest improvement, there were some with multiple terms, unlike those with opposite quality difference. This could suggest that our solution better adapted to multi-term queries.

Table 7.4: Entropy queries with largest AP differences (versus baseline).

(a) Highest difference (i.e., bigger improvement).

Query	AP Difference
red bu	0.804
marcel	0.775
denis	0.732
riga	0.685
campeonato nacional juniores	0.675
ramon	0.639
michael	0.635
arca	0.589
maca	0.589
dieg	0.545
lincoln	0.500
anselmo	0.499
ailton	0.467
adilson	0.445
guine	0.436
souza	0.433
liga i	0.422

(b) Lowest difference (i.e., bigger loss).

Query	AP Difference
champions	-0.542
diogo	-0.395
chi	-0.395
alan	-0.322
port	-0.318
mauro	-0.317
marcio	-0.310
marc	-0.306
paulo	-0.294
lusit	-0.285
rafa	-0.278
mina	-0.250
alex	-0.228
ala	-0.198
michel	-0.196
capi	-0.189
paraguay	-0.189

7.4 Summary

Our evaluation process consisted of comparing results obtained from different combinations of our proposed solution and the current production engine. For this, we extracted two query sets from the search logs, one with the most frequents queries (Frequent Query Set) and one with the most ambiguous ones (Entropy Query Set). To annotate queries with the correct answer, we had to resort to using clicks as a relevance proxy. For the Frequent Query Set, there was one correct answer, the most clicked entity, since they always had a high click share, averaging 85%. As for the Entropy Query Set, a majority was seldom found, hence all entities clicked were considered in a way to reflect graded relevance. In order to mitigate some possible threats that come with it, for evaluation, we used several distinct IR metrics, including MAP, MRR (as well as a modified weighted version), Success and NDCG. Moreover, for the Entropy Query Set, we also measured similarity to ideal rankings via measuring Spearman rank correlation coefficients, followed by hypothesis tests for significance.

Results obtained for frequent queries showed we were able to match the current system quality, which was already considered good. On the other hand, results for more ambiguous queries demonstrated the improvements provided by our solution, when considering user access patterns as a reference point. Both *PProd* and *PSum* strategies appeared to be the most promising, with

a small advantage for the former, when considering both of the tested query sets, as they had the most stable values across both query sets. In fact, stability and consistency was observed for results across metrics for different strategies, providing confidence on the reliability of obtained results.

Overall, these findings suggested that the proposed solution fulfilled its expectations. This was further corroborated by the fact that, for the Entropy Query Set, we possessed only part of the full correct result set, versus the baseline, which had access to full collections, and, therefore, all answers, and still managed to obtain performance enhancements.

Chapter 8

Conclusions

In this work, we targeted two distinct areas that are, nonetheless, ever more intertwined. The main problem tackled concerned a retrieval system in the form of a search engine. Information Retrieval (IR) is a research area closely tied to search tasks and, therefore, saw a renewed attention and focus with the World Wide Web. However, some search engines operate under unique circumstances. More specifically, their user base could be more restricted, as well as the collection it indexes, for its growth could happen in a semi-manually controlled fashion. Our focused project, zerozero.pt, a Portuguese sports website where search results were entities of different types (e.g., players, teams), fell under this category, presenting as a domain-specific search engine and, thus, demanding solutions fine-tuned to their context. The main identified problems lied within the lack of knowledge of user search patterns and of result quality for search beyond the most popular queried documents.

In order to analyze search behavior patterns, we performed a Query Log Analysis (QLA) process. QLA is an established research area with well-defined steps, benefits and limitations, as well as general analysis levels, i.e., term, query and session, that allowed for an easier comparison with findings from previous studies. Moreover, we presented a click level perspective seldom seem in other work. Previous work fell into one of two main categories: open Web, or general purpose, and domain-specific. Pre-processing for non-human interactions revealed a lower machine-originated activity pattern than previous findings by other authors in other domains.

Both term and session analysis showed classic Web search patterns, with users preferring shorter terms and having typically short search sessions. An exception was unique term percentage, which turned out to be very context specific, a pattern also observed when surveying other work.

Query analysis confirmed patterns observed in the term level perspective, namely an average caching effect and shorter lengths, with single term queries assuming wider presence. Moreover, there was a high new search ratio on a daily basis, i.e., of never seen before queries. Portuguese and Brazilian searchers amounted to over 90% of searches. Weekends, where games typically occurred, were the preferred periods to search. Daily query volume suffered a 25 to 50% drop that coincided with the impact of the COVID-19 pandemic on sports events. This led to some

behavioral changes, more specifically on hourly patterns and which entities users searched. Furthermore, we found that football matches caused localized search spikes that didn't span much beyond the timeframe of the event.

Finally, at click level, we observed that most clicks were on the top 10 results. Due to being a rare dimension in analysis, we presented some existing metrics for benchmark creation. Query click entropy analysis showed that shorter queries, often single names, were the most ambiguous. To measure user satisfaction, we proposed a metric by adapting the traditional TF-IDF retrieval formulation. Results showed how longer queries produced lower quality results in the current engine.

In order to propose enhancements to the search engine, we studied how these have evolved. More specifically, we saw how different search tasks emerged as a consequence of different needs and how they originated new indexing and retrieval strategies that could be applied to solve our problem. We opted to propose a Federated Search solution, where each collection corresponded to an entity type, and we also indexed previous search terms for a given document, a strategy shown to have high discriminative power. Each term had an associated relevance weight dependent on how frequently it was used not only to reach that, but also other documents (once again, a TF-IDF adaptation). These weights were then incorporated into the documents' relevance score when a match occurred. Four incorporation strategies were defined as a result of two binary variables: score update (linear scaling or linear combination) and previous search term frequency (raw value or log-smoothed). To build our prototype, we used Solr to create each collection and a Python Flask server as a middleware between Solr and the user in order to perform results merging across collections.

Our evaluation process then consisted of comparing results obtained between different combinations of our proposed solution and the current production engine. We used two query sets extracted from the search logs: one with the most frequent queries and one with the most ambiguous ones, defined by their entropy. To annotate queries with correct answers, we had to resort to using clicks as a relevance proxy. In order to mitigate possible caveats, we used several distinct IR metrics for evaluation, including MAP, MRR, Success, NDCG and similarity to ideal rankings via Spearman rank correlation coefficients.

Results obtained for frequent queries showed we were able to match the current system quality, which was already considered good. On the other hand, results for more ambiguous queries demonstrated the improvements provided by our solution. Raw search term frequency achieved better results than its counterpart (the difference was integration dependent, though, for some, it was close to 10%), with both linear scaling and combination providing the best results alongside it, with a slight advantage for the former (a maximum of two to three percentage points difference). In fact, stability was observed across metrics for different strategies, reinforcing confidence on the reliability of the obtained results. This showed how a single, uniform and integrated system was able to provide quality answers for a diverse set of information needs and queries.

8.1 Main Contributions

Our main contributions were also two-folded. First, we conducted a Query Log Analysis on a new domain that hadn't been explored before. With it, we were able to identify which common search patterns remained reasonably constant, no matter the context, and what unique characteristics defined this engine. We also were able to conduct specific period analyses, namely the impact of a pandemic on search behavior, that can, in the future, further help describe the worldwide impact of COVID-19 across several areas of our daily lives. Moreover, we presented a perspective that is seldom found in other studies based on the analysis of click interactions. Given their rarity, we proposed a set of simple metrics, including one created by us, that can be easily replicated by other researchers in their studies. Overall, we also hoped to have reinforced the importance of the process of QLA, not only to understand the user base, but also to identify possible enhancement areas in a retrieval system.

Regarding new strategies that can help benefit search results, we focused on the indexing and retrieval processes, though mainly on the former, and showed how previous work on individually-weighted search term indexing could be expanded. By successfully modelling the problem as a Federated Search instance, we implemented a solution that added a new signal to default Solr relevance and produced better results. Instead of using term weights as a substitute, we demonstrated that, by incorporating them as a complement to traditional relevance score, a search engine would be able to more easily highlight common access patterns for ambiguous queries. This was corroborated by effectively using production click-through information as a relevance proxy.

8.2 Future Work

This work could be continued in different ways. Regarding the Query Log Analysis process, a main interest would be to re-run the analysis with a query log that has a broader collection period, ideally enough months to cover a whole football season (typically mid-August to May). This would not only provide the chance to confirm results obtained here, but also to incorporate other types of metrics that measure how other phenomena, such as searches for specific entities, evolved throughout a season, and correlate them with key events. While our original intent didn't have such a scope, we wanted to cover a few months of interactions. However, we were unfortunately limited by the COVID-19 pandemic. Moreover, there are always analysis processes present in other studies that, ultimately, weren't replicated due to time constraints. In this case, it would be interesting to perform, as an example, manual query classification for a query log subset in order to obtain a more thorough distribution of queried entity types. While we used entity click information to this end, a manual process would also make it possible, for instance, to accurately detect entities that aren't directly searchable, like games. Other perspectives, such as a thorough user level analysis, by further exploring the influence of authentication in search, would also be an interesting path.

As for the search engine prototype, both implementation and evaluation could serve as a pillar for continued work. For the former, it would be interesting to experiment with other types of incorporation strategies beyond what was tested here. This could lead to a more systematic analysis in order to assess possible patterns concerning optimal strategy types. Another possible continuation includes replicating the behavior of the current system by favoring terms used more recently, so as to avoid wrongful bias of possible search term spikes. Regarding evaluation, quality measurement under real system usage (i.e., solution deployment to production) would further aid in testing the solution's quality. Moreover, it would allow the solution to have access to full collections. This way, it would be possible to perform, for example, A/B testing against a baseline, knowing that both operate under the same conditions.

References

[1] Sanae Achsas and El Habib Nfaoui. An Analysis Study of Vertical Selection Task in Aggregated Search. *Procedia Computer Science*, 148:171–180, 2019.

[2] Lada A Adamic and Bernardo A Huberman. Zipf's law and the internet. *Glottometrics*, 3(1):143–150, 2002.

[3] Maristella Agosti, Franco Crivellari, and Giorgio Maria Di Nunzio. Web log analysis: A review of a decade of studies about information acquisition, inspection and interpretation of user interaction. *Data Mining and Knowledge Discovery*, 24(3):663–696, 2012.

[4] Apache. Apache lucene - apache lucene core. http://lucene.apache.org/core/, 2019. [Online; accessed January 2020].

[5] Jaime Arguello. *Federated Search for Heterogeneous Environments*. PhD thesis, Carnegie Mellon University, 2011.

[6] Jaime Arguello. Aggregated search. *Foundations and Trends® in Information Retrieval*, 10(5):365–502, 2017.

[7] Jaime Arguello, Fernando Diaz, Jamie Callan, and Jean Francois Crespo. Sources of evidence for vertical selection. *Proceedings - 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009*, pages 315–322, 2009.

[8] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[9] Michael Barbaro and Tom Zeller Jr. A face is exposed for aol searcher no. 4417749. *The New York Times*, 2006.

[10] Steven M Beitzel, Eric C Jensen, Abdur Chowdhury, Ophir Frieder, and David Grossman. Temporal analysis of a very large topically categorized web query log. *Journal of the American Society for Information Science and Technology*, 58(2):166–178, 2007.

[11] Steven M. Beitzel, Eric C. Jensen, Abdur Chowdhury, David Grossman, and Ophir Frieder. Hourly analysis of a very large topically categorized Web query log. *Proceedings of Sheffield SIGIR - Twenty-Seventh Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 321–328, 2004.

[12] Tristan Blanc-Brude and Dominique L. Scapin. What do people recall about their documents? implications for desktop search tools. In *Proceedings of the 12th International Conference on Intelligent User Interfaces*, IUI '07, page 102–111, New York, NY, USA, 2007. Association for Computing Machinery.

[13] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, April 1998.

[14] A. Z. Broder and A. C. Ciccolo. Towards the next generation of enterprise search technology. *IBM Systems Journal*, 43(3):451–454, 2004.

[15] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. Efficient query evaluation using a two-level retrieval process. *International Conference on Information and Knowledge Management, Proceedings*, pages 426–434, 2003.

[16] Chris Buckley and Ellen M. Voorhees. Evaluating evaluation measure stability. *SIGIR Forum*, 51(2):235–242, August 2017.

[17] Vannevar Bush et al. As we may think. *The Atlantic*, 176(1):101–108, 1945.

[18] Stefan Büttcher, Charles LA Clarke, and Brad Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 621–622. ACM, 2006.

[19] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching distributed collections with inference networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, page 21–28, New York, NY, USA, 1995. Association for Computing Machinery.

[20] Jamie Callan. Distributed Information Retrieval. In *Advances in Information Retrieval*, pages 127–150. Kluwer Academic Publishers, Boston, 2005.

[21] Michael Chau, Xiao Fang, and Olivia R.Liu Sheng. Analysis of the query logs of a web site search engine. *Journal of the American Society for Information Science and Technology*, 56(13):1363–1376, 2005.

[22] Fabio Crestani and Ilya Markov. Distributed information retrieval and applications. In *European Conference on Information Retrieval*, pages 865–868. Springer, 2013.

[23] Qing Cui and Alex Dekhtyar. On improving local website search using web server traffic logs: A preliminary report. *Proceedings of the Interntational Workshop on Web Information and Data Management WIDM*, pages 59–66, 2005.

[24] DanBrown. csvquote: smart and simple csv processing on the command line. https://github.com/dbro/csvquote, 2018. [Online; accessed May 2020].

[25] DB-Engines. Db-engines ranking - popularity ranking of search engines. https://db-engines.com/en/ranking/search+engine, 2020. [Online; accessed January 2020].

[26] Federação Portuguesa de Futebol. Competições de futebol e futsal da fpf suspensas. https://www.fpf.pt/News/Todas-as-notícias/Notícia/news/26564, 2020. [Online; accessed April 2020].

[27] Chen Ding and Jin Zhou. Log-based indexing to improve web site search. In *Proceedings of the 2007 ACM symposium on Applied computing - SAC '07*, page 829, New York, New York, USA, 2007. ACM Press.

[28] Susan Dumais, Edward Cutrell, Jonathan J Cadiz, Gavin Jancke, Raman Sarin, and Daniel C Robbins. Stuff i've seen: a system for personal information retrieval and re-use. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 72–79, 2003.

[29] Elasticsearch. Elasticsearch reference [7.5]. https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html, 2020. [Online; accessed January 2020].

[30] Ronald Fagin, Ravi Kumar, Kevin S. McCurley, Jasmine Novak, D. Sivakumar, John A. Tomlin, and David P. Williamson. Searching the workplace web. In *Proceedings of the twelfth international conference on World Wide Web - WWW '03*, page 366, New York, New York, USA, 2003. ACM Press.

[31] Yi Fang, Naveen Somasundaram, Luo Si, Jeongwoo Ko, and Aditya P. Mathur. Analysis of an expert search query log. *SIGIR'11 - Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1189–1190, 2011.

[32] Marcus Fontoura, Maxim Gurevich, Vanja Josifovski, and Sergei Vassilvitskii. Efficiently encoding term co-occurrences in inverted indexes. *International Conference on Information and Knowledge Management, Proceedings*, pages 307–316, 2011.

[33] Marguerite Fuller, Liadh Kelly, and Gareth Jones. Applying contextual memory cues for retrieval from personal information archives. *PIM 2008 - Personal Information Management in conjunction with CHI 2008 Workshop*, 2010.

[34] Benjamin Ghansah and Sheng Li Wu. Distributed information retrieval: Developments and strategies. In *International Journal of Engineering Research in Africa*, volume 16, pages 110–144. Trans Tech Publ, 2015.

[35] Ben Gomes. Our latest quality improvements for search. https://blog.google/products/search/our-latest-quality-improvements-search/, 2017. [Online; accessed March 2020].

[36] Hatcher, Erik. Solr payloads. https://lucidworks.com/post/solr-payloads/, 2017. [Online; accessed May 2020].

[37] Davind Hawking. Challenges in Enterprise Search. *ADC '04 Proceedings of the 15th Australasian database conference - Volume 27*, 27:15–24, 2004.

[38] ISO.org. Iso 3166. https://www.iso.org/iso-3166-country-codes.html, 2020. [Online; accessed May 2020].

[39] Bernard J. Jansen. Search log analysis: What it is, what's been done, how to do it. *Library and Information Science Research*, 28(3):407–432, 2006.

[40] Bernard J. Jansen and Udo Pooch. A Review of Web Searching Studies and a Framework for Future Research. *Journal of the American Society for Information Science and Technology*, 52(3):235–246, 2001.

[41] Bernard J. Jansen and Amanda Spink. How are we searching the World Wide Web? A comparison of nine search engine transaction logs. *Information Processing & Management*, 42(1):248–263, jan 2006.

[42] Bernard J. Jansen, Amanda Spink, Chris Blakely, and Sherry Koshman. Defining a Session on Web Search Engines. *Journal of the American Society for Information Science and Technology*, 2006.

[43] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002.

[44] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02*, page 133, New York, New York, USA, 2002. ACM Press.

[45] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '05*, volume 51, page 154, New York, New York, USA, 2005. ACM Press.

[46] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: Development and comparative experiments part 2. *Inf. Process. Manage.*, 36(6):809–840, November 2000.

[47] Jinyoung Kim and W. Bruce Croft. Ranking using multiple document types in desktop search. *SIGIR 2010 Proceedings - 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, 2010.

[48] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[49] Arlind Kopliku, Karen Pinel-Sauvagnat, and Mohand Boughanem. Aggregated search. *ACM Computing Surveys*, 46(3):1–31, jan 2014.

[50] Reiner Kraft, Chi Chao Chang, Farzin Maghoul, and Ravi Kumar. Searching with context. In *Proceedings of the 15th international conference on World Wide Web*, pages 477–486. ACM, 2006.

[51] Anagha Kulkarni, Jaime Teevan, Krysta M Svore, and Susan T Dumais. Understanding temporal query dynamics. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 167–176, 2011.

[52] PengFei (Vincent) Li, Paul Thomas, and David Hawking. Merging algorithms for enterprise search. In *Proceedings of the 18th Australasian Document Computing Symposium*, ADCS '13, page 42–49, New York, NY, USA, 2013. Association for Computing Machinery.

[53] Yiqun Liu, Yupeng Fu, Min Zhang, Shaoping Ma, and Liyun Ru. Automatic search engine performance evaluation with click-through data analysis. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 1133–1134, New York, NY, USA, 2007. Association for Computing Machinery.

[54] Khalid Mahmood, Tore Risch, and Minpeng Zhu. Utilizing a NoSQL Data Store for Scalable Log Analysis. In *Proceedings of the 19th International Database Engineering & Applications Symposium on - IDEAS '15*, pages 49–55, New York, New York, USA, 2014. ACM Press.

[55] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[56] Behrooz Mansouri, Mohammad Sadegh Zahedi, Ricardo Campos, Mojgan Farhoodi, and Maseud Rahgozar. Understanding User's Search Behavior towards Spiky Events. In *Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18*, pages 1763–1769, New York, New York, USA, 2018. ACM Press.

[57] MarkLogic. Marklogic 10 product documentation. http://docs.marklogic.com/, 2020. [Online; accessed January 2020].

[58] Calvin N Mooers. *The theory of digital handling of non-numerical information and its implications to machine economics*. Zator Co., 1950.

[59] Michael Oakes and Yan Xu. A search engine based on query logs, and search log analysis at the university of sunderland. *CEUR Workshop Proceedings*, 1175, 2009.

[60] OWASP. Top ten web application security risks. https://owasp.org/www-project-top-ten/, 2017. [Online; accessed May 2020].

[61] Thomas A Peters. The history and development of transaction log analysis. *Library hi tech*, 11(2):41–66, 1993.

[62] MF Porter. An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, 14, 03 1980.

[63] Yves Rasolofo, Faïza Abbaci, and Jacques Savoy. Approaches to collection selection and results merging for distributed information retrieval. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, page 191–198, New York, NY, USA, 2001. Association for Computing Machinery.

[64] Rui Ribeiro. Characterization of Portuguese Web Searches. Master's thesis, Faculdade de Engenharia da Universidade do Porto (FEUP), 2011.

[65] Armin Ronacher. Flask documentation. https://flask.palletsprojects.com/en/1.1.x/, 2010. [Online; accessed May 2020].

[66] Mark Sanderson and W Bruce Croft. The history of information retrieval research. *Proceedings of the IEEE*, 100(Special Centennial Issue):1444–1451, 2012.

[67] Ralf Schenkel, Andreas Broschart, Seungwon Hwang, Martin Theobald, and Gerhard Weikum. Efficient Text Proximity Search. In *String Processing and Information Retrieval*, volume 4726 LNCS, pages 287–299. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[68] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples)†. *Biometrika*, 52(3-4):591–611, 12 1965.

[69] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, September 1999.

[70] Solr. Apache solr - features. http://lucene.apache.org/solr/features.html, 2019. [Online; accessed January 2020].

[71] Solr. The extended dismax query parser. https://lucene.apache.org/solr/guide/8_4/the-extended-dismax-query-parser.html, 2019. [Online; accessed May 2020].

[72] C Spearman. *"General Intelligence" Objectively Determined and Measured.* Appleton-Century-Crofts, East Norwalk, CT, US, 1961.

[73] Sphinx. Sphinx | open source search server. http://sphinxsearch.com/docs/sphinx3.html, 2018. [Online; accessed January 2020].

[74] Amanda Spink, Bernard J Jansen, Dietmar Wolfram, and Tefko Saracevic. From e-sex to e-commerce: Web search changes. *Technology*, 53(2):226–234, 2001.

[75] Amanda Spink, Dietmar Wolfram, Major BJ Jansen, and Tefko Saracevic. Searching the web: The public and their queries. *Journal of the American society for information science and technology*, 52(3):226–234, 2001.

[76] Splunk. Splunk enterprise - splunk documentation. https://docs.splunk.com/Documentation/Splunk, 2020. [Online; accessed January 2020].

[77] Andreas Staeding. List of user-agents (spiders, robots, browser). http://www.user-agents.org/index.shtml, 2011. [Online; accessed January 2020].

[78] Xiaoya Tang and Bryan Heidorn. The loss of domain knowledge in user search queries: A query log analysis of a botanical retrieval system. *Proceedings of the ASIST Annual Meeting*, 44, 2007.

[79] Tao Tao and Chengxiang Zhai. An exploration of proximity measures in information retrieval. *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'07*, pages 295–302, 2007.

[80] Sergio Duarte Torres, Djoerd Hiemstra, and Pavel Serdyukov. Query log analysis in the context of information retrieval for children. *SIGIR 2010 Proceedings - 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 847–848, 2010.

[81] Howard Turtle and James Flood. Query evaluation: strategies and optimizations. *Information Processing & Management*, 31(6):831–850, 1995.

[82] Sarah K. Tyler and Jaime Teevan. Large scale query log analysis of re-finding. In *Proceedings of the third ACM international conference on Web search and data mining - WSDM '10*, page 191, New York, New York, USA, 2010. ACM Press.

[83] Walter Underwood. Measuring search relevance with mrr. https://observer.wunderwood.org/2016/09/12/measuring-search-relevance-with-mrr/, 2016. [Online; accessed May 2020].

[84] UserAgentString.com. List of user agent strings. http://www.useragentstring.com/pages/useragentstring.php, 2018. [Online; accessed January 2020].

[85] Ellen M. Voorhees and Dawn M. Tice. Building a question answering test collection. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, page 200–207, New York, NY, USA, 2000. Association for Computing Machinery.

[86] Peiling Wang, Michael W. Berry, and Yiheng Yang. Mining longitudinal web queries: Trends and patterns. *Journal of the American Society for Information Science and Technology*, 54(8):743–758, jun 2003.

[87] Martin White. *Enterprise Search*. O'Reilly Media, Champaign, IL, USA, 2012.

[88] Martin White. Critical success factors for enterprise search. *Business Information Review*, 32(2):110–118, jun 2015.

[89] Martin White. Enterprise search. In *Making Search Work*, pages 113–126. Facet, 2018.

[90] Ryen White and Eric Horvitz. Cyberchondria: Studies of the escalation of medical concerns in web search. *ACM Trans. Inf. Syst.*, 27, 11 2009.

[91] Ryen W White, Ian Ruthven, and Joemon M Jose. The use of implicit evidence for relevance feedback in web retrieval. In *European Conference on Information Retrieval*, pages 93–109. Springer, 2002.

[92] Wikipedia. (exponential) moving average. https://en.wikipedia.org/wiki/Moving_average#Exponential_moving_average, 2020. [Online; accessed May 2020].

[93] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[94] Dietmar Wolfram, Amanda Spink, Jim Jansen, and Tefko Saracevic. Vox populi: The public searching of the web. *JASIST*, 52:1073–1074, 10 2001.

[95] Worldometer. Coronavirus update (live). https://www.worldometers.info/coronavirus/, 2020. [Online; accessed April 2020].

[96] Gui Rong Xue, Hua Jun Zeng, Zheng Chen, Wei Ying Ma, and Chao Jun Lu. Log mining to improve the performance of site search. *WISE 2002 - Proceedings of the 3rd International Conference on Web Information Systems Engineering Workshops*, pages 238–245, 2002.

[97] Gui-Rong Xue, Hua-Jun Zeng, Zheng Chen, Wei-Ying Ma, Hong-Jiang Zhang, and Chao-Jun Lu. Implicit link analysis for small web search. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 56–63. ACM, 2003.

[98] Hao Yan, Shuming Shi, Fan Zhang, Torsten Suel, and Ji Rong Wen. Efficient term proximity search with term-pair indexes. *International Conference on Information and Knowledge Management, Proceedings*, pages 1229–1238, 2010.

[99] Kwan Yi, Jamshid Beheshti, Charles Cole, John E. Leide, and Andrew Large. User search behavior of domain-specific information retrieval systems: An analysis of the query logs from PsycINFO and ABC-Clio's Historical Abstracts/America: History and Life. *Journal of the American Society for Information Science and Technology*, 57(9):1208–1220, jul 2006.

[100] Jin Zhou, Chen Ding, and Dimitrios Androutsos. Improving web site search using web server logs. In *Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research - CASCON '06*, page 22, New York, New York, USA, 2006. ACM Press.

[101] Huaiyu Zhu, Sriram Raghavan, Shivakumar Vaithyanathan, and Alexander Löser. Navigating the intranet with high precision. In *Proceedings of the 16th international conference on World Wide Web - WWW '07*, page 491, New York, New York, USA, 2007. ACM Press.

[102] Mingjie Zhu, Shuming Shi, Mingjing Li, and Ji Rong Wen. Effective top-K computation in retrieving structured documents with term-proximity support. *International Conference on Information and Knowledge Management, Proceedings*, pages 771–780, 2007.

[103] Mingjie Zhu, Shuming Shi, Nenghai Yu, and Ji Rong Wen. Can phrase indexing help to process non-phrase queries? *International Conference on Information and Knowledge Management, Proceedings*, pages 679–688, 2008.

Appendix A

Bot User-Agent List

Table A.1: Bot User-Agent regular expression list used for bot detection.

feed	bot	rss	^sapo
crawler	yahoo	libwww-perl	check_http
autoproxy	Apple-Pubsub	Jakarta	Microsoft URL
^Microsoft Office	vb project	bloglines	zend
WordPress	Atomic_email	^Mozilla/4.0$	^Mozilla/5.0$
AppEngine-Google	Mail.Ru	PostRank	NSPlayer
FINLY	VERSION_TABLE	Akregator	Azureus
utorrent	HTTP agent	System.Net.AutoWebProxyScriptEngine	OutlookConnector
ESS Update	OSSProxy	aol/http	NetworkedBlogs
PubSubAgent	SOAP	^w3af	charlotte
mediapartners-google	newsgator	bittorrent	Contacts
BTWebClient	WLUploader	MPFv	JNPR
checker	greatnews	Winhttp	Drupal - 37
DataCha0s	Apache-HttpClient	unchaos	DAP
activesync	cache	kevin	webcopier
Aberja	CE-Preload	Infoseek	Sitewinder
^java	PF:INET	plagger	python
^PHP	liferea	Ruby BlogzIce	^CFNetwork
ichiro	^Windows-Media-Player	Windows-Update-Agent	reaper
Twitturly	Bookdog	Referrer Karma	ia_archiver
findlinks	facebookexternalhit	CHttp	Kaspersky
Wget	Transmission	AltaVista	Reeder
reader	nutch	PuxaRapido	Sphider
!Susie	CheckLinks	vagabondo	agadine
research	Yandex	silk	larbin
l.webis	iTunes	spider	lwp
AppleSyndication	scoutjet	Microsoft-CryptoAPI	Rome Client
curl	Microsoft BITS	AdminSecure	stackrambler
Anonym	OpenCalaisSemanticProxy	vlc	validator
MicroMessenger	weborama-fetcher	AddThis.com	admantx

Appendix B

Stopword List

Table B.1: Stopwords list used.

a	as	aos	ao
como	com	da	de
do	dos	das	e
em	era	entre	la
meu	mais	me	nao
nos	na	no	nas
o	os	ou	onde
quem	qual	que	por
para	se	ser	sem
suas	sua	sob	sobre
uma	uns	umas	um

Appendix C

QLA Process

The QLA process adopted followed a strategy based on automation via Makefiles and Bash scripting, similar to what is done with several tools developed by community users. Below is the directory structure for the whole process.

```
Makefile
unique_queries.sh
click_entries.sh
popup_queries.sh
prep_scripts
    botlist.txt
    remove_bots.sh
    ...
analysis
    term
        output
        term_analysis.sh
        extract_terms.pl
        stopwords.txt
        ...
    query
    click
    session
```

There is a Makefile with a set of rules dedicated to each main step of the process, i.e., Cleaning, Preparation, and one for each level of analysis. While there are no direct inputs from the user, they are parametrized so that only a change in key variable values is necessary before re-running the process. By default, an *all* rule executes all the steps in the order Cleaning, Preparation, Analysis.

Listing C.1 shows an excerpt of the Makefile developed.

Listing C.1: QLA Makefile excerpt.

```
all: clean prepare term_analysis query_analysis click_analysis
    session_analysis
# Log files variables
COMPRESSED_LOG_FILE={input_here}
# Generic script variables
PREPARATION_SCRIPTS_DIR = prep_scripts
ANALYSIS_SCRIPTS_DIR = analysis
# Term level variables
TERM_SCRIPTS = ${ANALYSIS_SCRIPTS_DIR}/term
TERM_DICT_FILE = term_dict.txt
# Other variables ...

prepare:
   gzip -dck ${COMPRESSED_LOG_FILE} | tail -n +2 | csvquote | cut -d ',
       ' -f 2,5,6,8,9,10,13,14,17,18,19,20\
      | sh ${PREPARATION_SCRIPTS_DIR}/remove_bots.sh | sh ${
         PREPARATION_SCRIPTS_DIR}/filter_strange_queries.sh | sh ${
         PREPARATION_SCRIPTS_DIR}/filter_empty_queries.sh\
      | sh ${PREPARATION_SCRIPTS_DIR}/sort_by_sessid_datetime.sh | sh $
         {PREPARATION_SCRIPTS_DIR}/add_session_id.sh | sh ${
         PREPARATION_SCRIPTS_DIR}/remove_long_sessions.sh\
      | sh ${PREPARATION_SCRIPTS_DIR}/convert_charset.sh > ${
         LOG_POST_PREPARATION}

term_analysis:
   cat ${LOG_POST_PREPARATION} | sh unique_queries.sh | cut -d ',' -f
       10 | perl ${TERM_SCRIPTS}/extract_terms.pl > ${TERM_SCRIPTS}/
       output/${TERM_DICT_FILE}
   sh ${TERM_SCRIPTS}/term_analysis.sh ${TERM_DICT_FILE} output

# Similar rules with pipelined processes for query, session and click
    analysis
```

Each step consists of one or more pipelined processes to execute all the steps necessary. In the case of Preparation, the end result is a CSV file with a processed log, to be consumed by the Analysis rules. In the case of Analysis, each rules may do some preprocessing, before calling an independent script that will run all metrics at that level. In the case of Term analysis shown, it first extracts unique queries only, then selects the query field, and finally builds the term frequency dictionary to serve as input for an analysis script.

Independent analysis scripts are then grouped inside the analysis level's dedicated folder and

used in a new set of pipelined processes to calculate each designated metric. It is here also that helper analysis scripts (e.g., processing Perl scripts or plotting R scripts) are stored. Outputs are saved to an *output* directory that is deleted every time the Clean rule is ran. In the case of plots, the correct directory is passed to the R programs that output them. In the case of text files, redirection (>) is used. Listing C.2 shows an excerpt of the Query analysis script, demonstrating the usage of variables to dictate current directory, output directory, text files with parametrized input (for stopwords and football matches for specific period analysis) and some metrics.

Listing C.2: Query analysis script excerpt.

```
#!/bin/bash
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 && pwd )"
OUTPUT="$DIR"/output
QUERY_DICT_FILE="$OUTPUT"/query_dict.txt
STOPWORDS_FILE="$DIR"/stopwords.txt
GAMES_FILE="$DIR"/games.txt

# Pre calculate query (with processing) frequency dictionary to serve as
    input for some of the metrics
cut -d ',' -f 8 "$OUTPUT"/$1 | sh "$DIR"/normalize_queries.sh | sort |
    uniq -c | sort -nr | awk -F ' ' 'BEGIN{OFS=" "} {freq=$1; $1=""; print
    substr($0,2) "--&&&&--" freq}' > "$QUERY_DICT_FILE"

# Top queries
head -n 25 "$QUERY_DICT_FILE" > "$OUTPUT"/top_queries.txt

# General numerical query statistics
cat "$QUERY_DICT_FILE" | R -f "$DIR"/query_stats.R --args "$STOPWORDS_FILE
    " | grep -E "^\[1\]" | sed -r 's/\[1\]\s//' > "$OUTPUT"/query_stats.txt

# New searches ratio by day
cut -d ',' -f 1,8 "$OUTPUT"/$1 | perl "$DIR"/moving_unique_queries.pl > "
    $OUTPUT"/new_searches_daily.txt

# Query geographical distribution
cut -d ',' -f 7 "$OUTPUT"/$1 | sh "$DIR"/country_distribution.sh > "
    $OUTPUT"/country_distribution.txt
```

Adding more metrics is, then, a matter of creating the helper scripts and adding the respective pipeline to this analysis script.