EXPLORING THE TERRIER INFORMATION RETRIEVAL PLATFORM FOR WEB SEARCH OF DOCUMENTS WRITTEN IN MACEDONIAN

Vasil Vangelovski
Faculty of Computer Science and Engineering

Skopje, Macedonia

Sonja Gievska
Faculty of Computer Science and Engineering

Skopje, Macedonia

ABSTRACT

Terrier is a modular and scalable platform for rapid development of Information Retrieval (IR) systems. This paper presents a short overview of the Terrier architecture and describes ways in which it can be extended for more effective indexing and searching of documents written in Macedonian language. Although Terrier supports out of the box search in a few non-English languages, the Macedonian language poses some specific challenges, especially when search of Web content is involved. An integrated search platform is developed for the purpose of this research extending the text retrieval engine with a more advanced content filtering capabilities. Some of the proposed methods can be easily applied to other non-English languages.

I. INTRODUCTION

The ever-increasing quantities of web content require versatile, scalable and effective search solutions to be integrated into applications and websites. While there is a number of established commercial and open source search solutions most of them are focused on search of English text. Text indexing and searching is closely related to the grammatical rules of the language under investigation hence the rules of the English language are rarely appropriate for other languages.

Search engines use stemming to reduce words to their root, but in Macedonian a stem of the word may be a non-existing word or a word with a different meaning. In addition, no fixed number of letters subtracted from the end of the word guarantees a valid stem.

Web content written in Macedonian, especially user-generated content poses even more challenges when designing an information retrieval system because of the different ways text is written. Specifically, the official alphabet for writing in Macedonian is Cyrillic. Because of historical reasons (lack of support for Macedonian UTF layouts in earlier operating systems) and practical reasons (the need to switch between keyboard layouts) users often write Macedonian text using only ASCII characters by loosely transcoding from their Cyrillic phonetic counterparts. For example the letter **ц** is written as the Latin letter **c** and some letters are written using a combination of Latin letter and often in different ways for example the letter **ш** is often transcoded to **sh** or **sch** or simply **s**. The style of entering search keywords varies between users as well.

Terrier, TERabyte RetrIEveR [1], is a high-performance search engine that allows a rapid development of large- scale retrieval applications, by providing a comprehensive, flexible, robust and transparent platform for research and experimentation in IR. The development of the Terrier platform originated at the University of Glasgow with a primary goal to facilitate research into Web search, but has since been extended to include other applications and is available as open source software. It includes a desktop search application as well as a web search interface, although the platform can be used as a Java library to develop custom applications that utilize its text search facilities.

The architectural overview of the Terrier platform is followed by the discussion of the proposed solutions for some of the aforementioned problems. The to Terrier as a search engine for user-generated web content in Macedonian applied as an integral part of a real-world web application containing a real collection of user-generated text.

II. OVERVIEW OF THE IR TERRIER PLATFORM

A. *Indexing*

The Terrier indexing process is divided into four stages:

- Handling of a corpus of documents
- Handling and parsing of each individual document
- Processing the terms from and individual document
- Building the index data structures

Modularity of the indexing components is achieved by enabling the addition of plugins that can alter the behaviour at each stage [2]. For example, indexing a collection of documents stored on a FTP server would require an implementation of a new collection of plugins that connects to the FTP server and processes a stream of documents stored on the server side. Similarly, a support for a new document format could be added (besides the ones already supported) by implementing a document parser for the particular format and included as a plugin in the second stage.

Each term extracted from a document is defined by three properties: the actual string representation of the term, the position in the document where it occurs and the fields (zones in the document) where it occurs. With such representation terms are passed through a term pipeline, which allows each term's property to be transformed in various ways. The most common ways in which terms need to be transformed in a typical full text search application are stemming and removal of stop words. Terrier includes implementations for two variants of the Porter stemming algorithm, and a stop words removal for a few different languages.

©2013 Faculty of Computer Science and Engineering

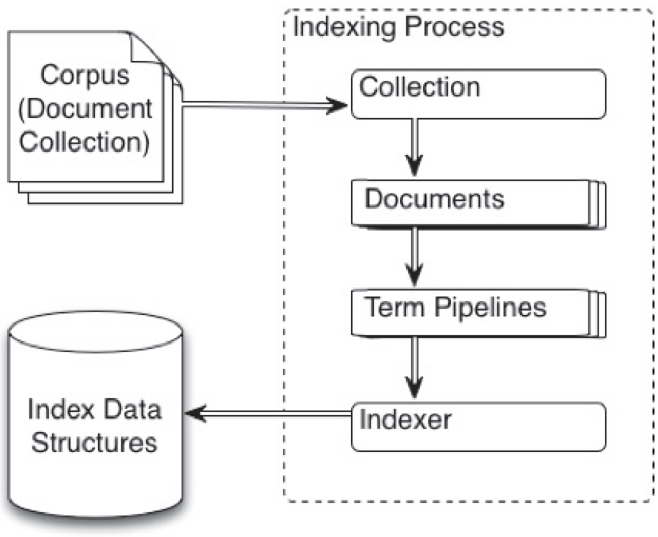

Figure 1: Terrier Indexing Process.

The indexer is the last component in the term pipeline. It is responsible for constructing the index using the appropriate data structures. A Terrier index consists of four main data structures:

- Lexicon - Stores the term, it's unique id number, global statistic for the term's frequency.
- Inverted Index - Stores the document unique id number and the term frequency for each term in a particular document. It can also store encoding of positional information for each term in the document.
- Document Index - Stores a document number, the length of the document (number of tokens) and the offset of the document in the direct index.
- Direct Index - Stores the terms and term frequencies of the terms found in each document. The main objective for the direct index is to facilitate efficient query expansion, in addition to being useful when clustering a collection of documents.

B. *Retrieval*

The Terrier retrieval process is outlined in fig. 2. [2] [3]

The core functionality in a retrieval system is matching documents to queries and assigning them scores that represent their relevance to the query. Terrier offers a number of implemented matching models for calculating a document score, including a novel DFR - divergence from randomness model [4]. A developer is free to choose from any of the weighting models or provide their own implementation of a weighting model.

There is set of term score modifiers for changing the initial weighted term score. For example, a *TermInFieldModifier* can be used to ensure that terms are found only in certain fields in a document, while a *FieldScoreModifier* can be used to increase the score of documents that have a term in a certain field. Document scores can be modified by employing document score modifiers, such as the *PhraseScoreModifier*, which is used to remove documents that do not contain the query terms. Document score modifiers are also very useful for applying query-independent score modifications, such as modifying the document scores based on the relevance of the documents pointed by the hyperlinks embedded in the document.

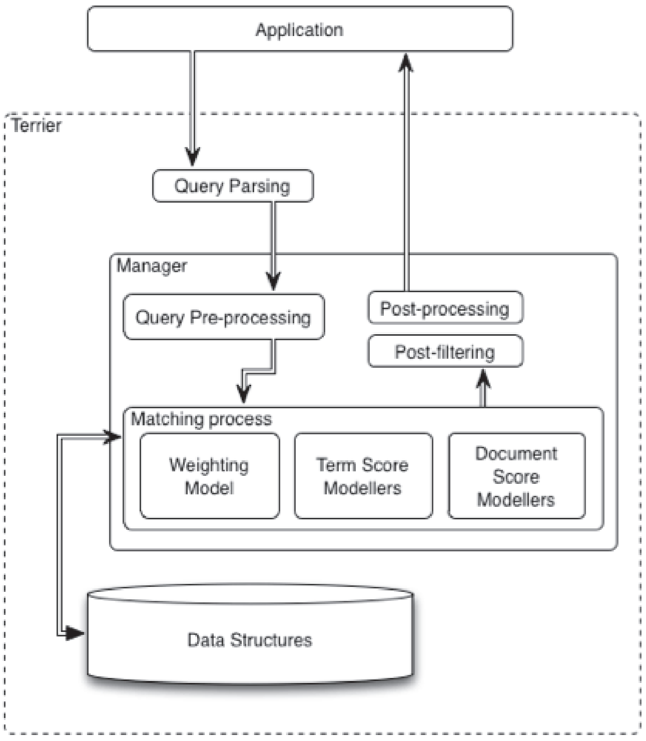

Figure 1: Terrier Retrieval Process.

Post-filtering of the scored documents returned by the matching process can be applied to remove any documents that don't satisfy certain conditions, for example the number of documents from the same source or author can be reduced to increase the diversity of the search results.

In addition to the modular and flexible retrieval architecture, Terrier includes a flexible and powerful query language [4]. The user of the retrieval system is allowed to augment the query with special operators in order to customize the results to better suit their need, for example:

- +(t1 t2) specifies that only documents having both terms are required
- +t1 ~t2 specifies that documents need to contain t1 and do not contain t2
- t1^4.3 t2^1.2 will set the weights of t1 and t2 to 4.3 and 1.2, respectively; suitable for specifying relative relevance of each term for a particular search operation.

Terrier includes an automatic query expansion facility, by extending the query with terms related to the terms used in a query. This method of query expansion can be further modified to suit particular application needs.

III. Application

A. Introduction

User-generated content on the web such as comments, forum posts, status updates etc. poses additional challenges when designing a text indexing and search platform. Specifically, a large portion of the content is often poorly formatted, grammatically incorrect, contains spelling mistakes and slang words or ambiguous abbreviations. The web content written in Macedonian is often based on different keyboard layouts either loosely transcoded into ASCII characters from Cyrillic or uses a combination of both. Because of these reasons test data from collections of formal documents such as Wikipedia articles is inadequate when assessing the capabilities of a web content IR system.

We have designed our solution as an integral part of a working classified ads portal with a majority of Macedonian users [5]. The portal is using a search solution based on the Solr [6] search engine in production, which conveniently serves as a reference for the performance of our Terrier-based design.

Classified listings in the application are entered within separate categories and subcategories, a purpose for the ad and a geographical region. Users can browse the content of the site according to such categorization. The search interface allows the users to further filter their search results by selecting one or multiple categories and/or regions.

The search solution used in the production is a separate server process, which accepts new content for indexing and query requests through an HTTP REST interface. When the search service is initialized, the index was built completely. With each additional listing posted to the site, the index is updated by sending a request to the search process with the content of the posted listing. Search queries entered in the application's interface are then passed as a request to the search service; the returned results are reformatted before presenting to the user. The actual indexing and retrieval processes are performed within one stage, the search process.

To accommodate our architectural requirements, the Terrier-based solution is an HTTP process that serves as an interface wrapper around the Terrier platform, providing an interface to our application consistent with the one used during production. The details of the implemented extensions related to both, the indexing and retrieval process of Terrier, are discussed in the following subsections.

B. Extension to the Indexing Process

To speed up the process of rebuilding the full document index we have created a separate **document collection** implementation, which indexes the entire content of the site by reading directly from the main relational database. For this representation of the content items, a new **document format** was implemented. Because our application relies on updating the search index with XML representation of new content, a separate document format was implemented specifically for this purpose.

Our extensions to the term pipeline were realized by re implementing the **tokenizer, stop words removal** and **stemmer** specifically for Macedonian. The stop words removal process removes stop words written in Cyrillic, but also relies on transcoding Latin stop words as they are often short and can rarely be transcoded ambiguously.

Our implementation of a stemmer is based on the existing porter stemmer for the Russian language [7] because it uses a subset of the Cyrillic alphabet similar to Macedonian. This stemmer relies on first transcoding any Cyrillic words to their Latin representations by a standard transcoding scheme and then applying the stemming algorithm to the transcoded content. Words written in Latin are stemmed unmodified. The calculated suffixes from the words are then matched against the original representation of the word to provide the final stem in order to keep the original encoding of the text for indexing purposes. Even though porter-based stemming algorithms are not very effective for Russian or Macedonian, we have decided to use such an implementation since it was already available in Terrier and required only few modifications to the stemming rules.

Our first modification to the actual indexer was altering the indexing process in such a way that two versions of each document were stored in the index, one for the original document and one for the transcoded version. While we are aware of the fact that this effectively doubles the size of the stored index structures, the textual content that is indexed per document is rarely longer than 100 words, which makes storage considerations of a lesser for this application. This eliminates the need to do any complicated query expansion in the retrieval process and solves the problem of differently formed queries within a phase, namely during the indexing process.

When searching for listings, users usually add filters to their query such as a category, subcategory and/or a region. One way to implement the filtering was to process the results returned from the search platform by running additional queries against the database using the category and region filters. However, this would have added to the overall amount of data transferred between the search service and the application and increased the overhead because of the database queries. By observing the habits of users of the application, we have also discovered that users expect to be able to input the category or region in the search query instead of checking the associated checkbox, for example they would enter “Fiat Bravo in Skopje” and expect the included geo location to be applied as a filter. A fully integrated search solution was proposed by incorporating the filtering capabilities during the indexing and retrieval process itself. A separate index for categories and regions were added alongside the content index. The title and content terms are stored in a standard content index, while documents are indexed by category and region in a separate index within Terrier. It resulted in a more effective chained indexing operation, by indexing the content and the title followed by updating the category index.

C. Extension to the Retrieval Process

Our extension to the retrieval process was mainly directed toward extending the query parsing, query expansion and document ranking components of the Terrier retrieval process. A custom query parser in Terrier that accepts queries in XML format from the application was designed to facilitate the filtering capabilities of the user interface. The user query and the selected filters (if any) are presented as separate entities. The user query was complemented with its transcoded version by tokenizing the text query and producing a transcoded version of each token. If the original token was entered in Cyrillic it was transcoded in Latin and vice versa.

The query text was also matched against the entries within the category index to detect the query terms that specify a category or a region. When a category filter preference was discovered within a query, it was regarded as a non- exclusive preference; each filter preference specified within the application interface is regarded as an exclusive filter preference. The approach allows the system to adjust to both cases, especially preventing degrading search performance when the name of a category or a region have a meaning as a search term that should be accounted for as opposed to considering it as a filter. In practice, whenever the query model contains exclusive filters, the results are matched against the category filter and the documents that do not match the filters are removed from the result list. Conversely, when a filter is considered non-exclusive, the category index is used only to affect the document ranking in the resulting list. Documents that match the non- exclusive filters are given a score boost for each filter they match.

IV. EVALUATION

Traditionally information retrieval systems are evaluated by using them to index a prepared test collection of documents, which is later searched and evaluated by comparing the ranking of the results against a prepared list of users' rankings. The process enables engineers to establish quantifiable metrics for evaluation that can be determined in an automated manner. Considering that ranked training collections of Macedonian content are not available, we have decided to conduct a user survey to examine the performance of the system in terms of its usability; improving the user satisfaction was the main objective for extending the Terrier architecture.

A group of seven volunteers participated in the survey. The participants were asked to select a list of 20 queries from the list of 100 most-frequently used queries according to the system logs. The users were presented with two URLs, one for each instance of the application; the mapping between URL and a particular version was not revealed to the participants. They were asked to enter each of the queries in both systems, compare the top ten results and log their opinion in terms of the relevance of the results. In order to be able to draw conclusions on how the system performed with regards to the sensitivity to the encoding of the entered query the users were asked to test the system by entering the query in Cyrillic and a version loosely transcoded in Latin.

There was between-subject variance of less than 5% and the overall cumulative results could be summarized as follows:

- When the search query was entered unmodified without any filters, both systems performed almost the same; 78% of the answers were undecided.
- For the queries transcoded to Cyrillic, 63% of the users were in favor of the Terrier-based system, while 28% were in favor of the original system.
- When the queries were transcoded to Latin, 57% of the users were in favor of the Terrier-based system as opposed to 24% in favor of the original system.
- When a category or geographical region was added to a query, 97% were in favor of our Terrier-based system, remaining 3% were in favor of the original system.

V. CONCLUSIONS AND FUTURE WORK

This research discusses the challenges and possible solutions for effective information retrieval of user-generated web content written in Macedonian. The Terrier architecture was used as a basis for an integrated search solution, which extends the retrieval system with filtering and ranking capabilities.

An exploratory study was conducted to confirm the extent of the hypothesized qualitative gains of the proposed architecture. Our future research efforts are directed toward comparative analysis of the various text retrieval models supported by Terrier for the context under our investigation, especially in terms of using more quantifiable metrics such as precision and recall.

REFERENCES

[1] Terrier IR Platform - http://terrier.org/

[2] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C Lioma, "Terrier: A High Performance and Scalable Information Retrieval Platform", in *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval* , 2006. Seattle, Washington, USA.

[3] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and D. Johson, "Terrier Information Retrieval Platform", in *Proceedings of the 27th European Conference on Information Retrieval,* 2005.

[4] I I. Ounis, C. Lioma, C. Macdonald and V. Plachouras, "Research Directions in Terrier: a Search Engine for Advanced Retrieval on the Web", in *Novatica/UPGRADE Special Issueon Next Generation Web Search*, Vol. 8, No. 1, 2007, pp. 49—56.

[5] oglasuva.me - http://oglasuva.me/

[6] Apache Solr - http://lucene.apache.org/solr/

[7] Snowball stemmer for Russian - http://snowball.tartarus.org/algorithms/russian/stemmer.html