Error tracking in search engine development

Swapnil Chaudhari[1] *Arjun Atreya V*[1] *Pushpak Bhattacharyya*[1] *Ganesh Ramakrishnan*[1]
(1) Department of CSE, IIT Bombay
{swapnil, arjun, pb, ganesh}@cse.iitb.ac.in

ABSTRACT

In this paper, we describe a tool that allows one to track the output of every module of a search engine. The tool provides the ability to perform pseudo error-correction by allowing the user to modify these outputs or tune parameters of the modules to check for improvement of results. Often it is important to see if certain surface level changes can help in the improvement of the result quality. This is crucial since it saves the immediate need to make changes in the system in terms of resource updation or development efforts. We describe query processing pipeline in sufficient detail and then show the efficacy of our tool for an example in Marathi along with giving a thorough error analysis for the example considered. We believe this paper will establish that such a tool is of significant importance for instant detection and correction of errors along with giving the readers an idea on how to develop the same.

KEYWORDS: INFORMATION RETRIEVAL, TRACKER, ERROR TRACKING, ERROR ANALYSIS TOOL

1 Introduction

Information retrieval refers to searching relevant documents satisfying the user information need. User information need is typically captured in the form of a query. A search system consists of two parts viz. offline processing and online processing.

Offline processing mainly consists of two parts – crawling and indexing. In crawling, documents from the web are fetched and stored. These documents have to be parsed and stored in optimal way in terms of both storage space and search time. For efficient retrieval of documents, an inverted index of terms in the documents is created.

Online processing consists of converting the user's information need in a format which facilitates the matching of query terms with terms in documents. Query expansion using feedback or thesaurus, semantic search, *etc.* are different ways of capturing user information need. A naive way to capture information need is to consider query terms as representative of user information need. These terms in the query have to be processed before they can be used to search documents. Processing of terms involves stop word removal, stemming and query formulation. Some of the search engines also do named entity recognition, multi word recognition or word sense disambiguation to enhance query processing. This processing is done in the form of a pipeline where output of one stage is fed as input to the next stage.

Most of these modules need language based resources and processing. For example, named entity recognition requires applying machine learning techniques on a large corpus and extracting named entities out of it. The output of named entity recognition engine is used as dictionary for searching named entities in the query. All such modules cannot generate an exhaustive resource list and are vulnerable to errors. Errors in each module degrade the overall performance of the system. Evaluation forums like TREC (TREC, 1992), CLEF (CLEF, 2000), NTCIR (NTCIR, 1999) and FIRE (FIRE, 2008) provide platform to evaluate the system performance in terms of precision, recall, MAP value, *etc*. However, these measures indicate end to end performance of the system and do not evaluate performance of individual module in the system. Since the architecture is pipelined, the error propagates and multiplies. In such architecture, tracking the root cause of error is important. To facilitate this *tracker* was developed.

Tracker is a tool which captures the input and output information of each stage of the pipeline and displays it to the assessor. This helps in identifying the root cause of the error. A relevance judgement tool is integrated in tracker to aid storing relevance judgments for each query.

The roadmap of the paper is as follows. We describe query processing pipeline in the next section with an example case of Marathi. In section 3, we look at different types of errors that can occur in a search engine. Then we discuss at different functionalities in Tracker in section 4 followed by implementation details in section 5. Tracker was used by more than 100 developers and assessors across the country. Section 6 covers these user experiences about tracker.

2 Query processing pipeline

Figure 1 illustrates query processing pipeline. A query given in Marathi is

मुंबईमधील किंवा रायगडमधील सुंदर राष्ट्रीय उद्यान

mumbaimadhil kiva raaygadmadhil sundar raashtriya udyaan *in Mumbai or in Raigad beautiful national park*

beautiful national park in Mumbai or Raigad

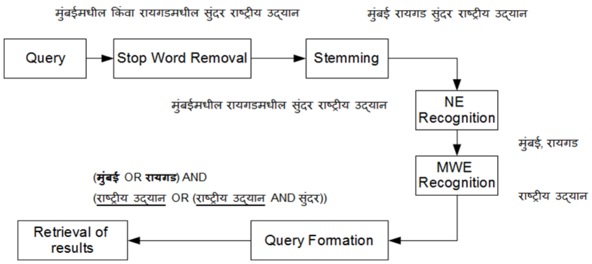

मुंबईमधील किंवा रायगडमधील सुंदर राष्ट्रीय उद्यान
मुंबई रायगड सुंदर राष्ट्रीय उद्यान
Query
Stop Word Removal
Stemming
मुंबईमधील रायगडमधील सुंदर राष्ट्रीय उद्यान
NE Recognition
मुंबई, रायगड
MWE Recognition
राष्ट्रीय उद्यान
(मुंबई OR रायगड) AND
(राष्ट्रीय उद्यान OR (राष्ट्रीय उद्यान AND सुंदर))
Retrieval of results
Query Formation

FIGURE 1- Query Processing illustrated with Marathi

The stop word removal module removes stop words by doing a dictionary look up. In this query किंवा (*kiva*) (*or*) is a stop word in Marathi which gets removed in this stage. The resultant query after removing stop word is मुंबईमधील रायगडमधील सुंदर राष्ट्रीय उद्यान (*mumbaimadhil raaygadmadhil sundar raashtriya udyaan*). This is given as an input to the stemmer.

The Marathi stemmer (Bapat *et. al*, 2010) is implemented as a Finite State Transducer in which we specify a word as a sequence of legal morphemes. Each morpheme corresponding to the root word is called the stem and this stem possesses features which are extracted by the means of a morphological parser. In this case, the suffix मधील (*madhil*) (*in*) is attached to words मुंबई (*mumbai*) (*Mumbai*) and रायगड (*raajgad*) (*Raigad*). This suffix is removed by the stemmer and the words मुंबईमधील (*mumbaimadhil*) (*in Mumbai*) and रायगडमधील (*raaygadmadhil*) (*in Raigad*) are reduced to their root forms मुंबई (*mumbai*) (*Mumbai*) and रायगड (*raaygad*) (*Raigad*) respectively. The resultant query is मुंबई रायगड सुंदर राष्ट्रीय उद्यान (*Mumbai raaygad sundar raashtriya udyaan*). The Marathi stemmer stems the words with an accuracy of 95 percent. These stemmed words are used by the named entity recognition module to detect named entities in the query.

The named entity recognition module detects named entities by searching through a dictionary of named entities. This dictionary is pre-computed from a large corpus. In this query, मुंबई (*mumbai*) (*Mumbai*) and रायगड (*raaygad*) (*Raigad*) are two named entities. After detecting named entities, multi-words in the query are detected.

The multi-word detection module takes an n-gram window to match the query terms against a dictionary of multi-words. This dictionary is precompiled from a large corpus. In this query, राष्ट्रीय उद्यान (*raashtriya udyaan*) (*national park*) is a multiword. After detecting the important terms in the query like named entities and multi-words, we form a Boolean query. The terms in the query are given appropriate boosts based on their importance. The results retrieved are then presented to the user. The quality of these set of results is a function of the accuracy of individual modules. Diagnostics of each module is called for at this stage.

3 Error analysis

Each module can contribute to error and affect the overall performance of the system. The following section describes the different kinds of errors that can occur in each stage of pipeline.

3.1.1 Types of errors and their impact on performance

In the stop word removal stage, errors can be due to a stop word not being detected by the module. This can boost non-relevant results to the top of ranked list because of high count of stop word content in it. Another possible error is wrongly detecting an important word as stop word and removing it from the query. Stemming involves two kinds of errors viz. Wrong stem and no stem. A wrong stem may be due to over-stemming or under stemming. Wrong stem results in change in meaning of the word used in the query. This can cause topic drift in results.

E.g. Consider the query

गुजरातचे लोक

gujaraatche loka

of Gujarat people

People of Gujarat

In this case, over stemming गुजरातचे (gujaraatche) (of Gujarat) will give root गुजर (gujar) (Gujar) instead of correct root गुजरात (gujarat) (Gujarat). गुजर (gujar) (Gujar) is a caste while गुजरात (gujarat) (Gujarat) is a state name. The query formed after stemming is (gujar people) (Gujar people). Instead of getting information about people of Gujarat, the user will get results related to people belonging to Gujar caste.

If the stemmer is not able to stem the word, it may return the original query term. In both cases, the error in stemming gets propagated to further stages. An error in stemming may cause error in detecting named entities and multi-words. In the above example, if गुजरातचे (gujaraatche) (of Gujarat) is not stemmed, then named entity गुजरात (gujarat) (Gujarat) will not be recognized. If a named entity is not recognized, we may lose the information about importance of that word. Similarly, if a multiword is not recognized, then information about importance as well as proximity of those query terms is lost. For example, consider the multi-word query राष्ट्रीय उद्यान (*raashtriya udyaan*) (*national park*). In this, if multi-word identification fails, then राष्ट्रीय (*raashtriya*) (*national*) and उद्यान (*udyaan*) (*park*) will be searched as two different entities. The word राष्ट्रीय (*raashtriya*) (*national*) can match with documents containing terms like राष्ट्रीय सीमा (*raashtriya seema*) (*national border*), राष्ट्रीय संस्था (*raashtriya sansthaa*) (*national institute*), राष्ट्रीय संग्रहालय (*raashtriya sangrahaalay*) (*national museum*), *etc.* and retrieve irrelevant results.

An error in a module may be corrected by adding resources or might require re-engineering to solve the issue. Changing the module for each error and retesting the system after change is quite time consuming. One way to analyze this problem is pseudo error correction. This involves correcting the output of a particular module temporarily without making a change in the module for detecting errors in further modules. To facilitate monitoring the outputs of individual modules and pseudo error correction, *tracker* was developed. As per our knowledge, no such tool for a search engine exists.

4 Tracker

Tracker is an error analysis tool developed to assist developers and assessors to analyze each module of a search engine for errors and tune the system parameter to find the best parameters that suit the system. Tracker captures input and output of each module for a query and displays them to the assessor. The assessor can then manually judge the outputs as correct or incorrect. This helps in detecting errors in modules.

For example, while processing the query मुंबईमधील किंवा रायगडमधील सुंदर राष्ट्रीय उद्यान (*mumbaimadhil kiva raaygadmadhil sundar raashtriya udyaan*), let us assume that stemmer is not able to remove the suffix मधील (*madhil*) (*in*) from words मुंबईमधील (*mumbaimadhil*) (*in Mumbai*) and रायगडमधील (*raaygadmadhil*) (*in Raigad*). As a result, the named entity module will not be able to detect मुंबई (*mumbai*) (*Mumbai*) and रायगड (*raaygad*) (*Raigad*) as named entities. By looking at the output of stemming stage, the assessor can conclude that stemmer should be further improved. However, even if we correct the stemmer, we are not sure whether named entity module will detect मुंबई (*mumbai*) (*Mumbai*) and रायगड (*raaygad*) (*Raigad*) as named entities.

To avoid delay in the detection of errors in subsequent modules, *Tracker allows assessor to replace the output of a particular module with the correct output without actually modifying the module*. Once the assessor submits the query again after modifying the stemmer output, the new corrected output will override the existing output and will be fed as input to next stage. This can be done incrementally to detect errors in all the modules. While doing this, tracker stores information about past changes and masks all the outputs of the modules which assessor wish to change. This information is stored till the assessor wish to clear it for that query. Let us see some additional capabilities in tracker which makes development and bug detection faster.

4.1.1 Capabilities of Tracker

A relevance judgement tool[1] is integrated with the tracker which allows the user to perform relevance judgement on a query. These judgements are stored in a database for future reference. The relevance judgements are pre-populated on the interface for subsequent firing of the same query. The result set for a query may change because of change in a module's output. The assessor has to perform relevance judgements only for those URLs whose relevance judgement was not stored earlier. This saves time of re-judging the same page for a query. The tracker automatically calculates precision values at rank 5 and 10 using these judgements. These figures help the assessor in analyzing the effect of current change in module outputs on the precision values of the result set.

Tracker also maintains a revision history for each query fired by the assessor. This history captures the changes done in different modules and corresponding precision values. This helps in summarizing what kind of changes can help to improve the system as a whole. Consider a scenario where the system was tested for around 100 queries. Out of these queries, 60% of the queries showed improvement after changing the output of stemmer. 20-30% showed improvement after modifying named entity recognition output and the rest didn't show a significant change in results after modifying any output. In this case, stemmer and named entity

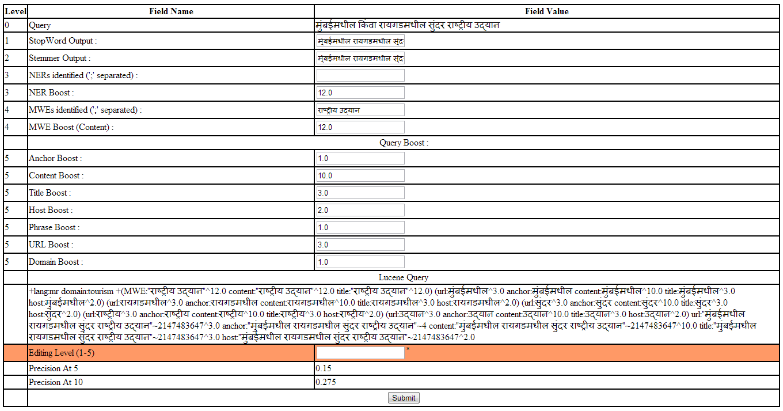
[1] A relevance judgement tool is an interface to help an assessor mark a retrieved URL as relevant, partially relevant, link-relevant, irrelevant and error

recognition modules should be improved. Such data can help prioritize tasks for development and help improve the system faster.

5 Implementation Details

Tracker is developed in Java. Tracker has a web interface which is linked to the results page of the search engine. The system maintains login information of the assessors to store relevance judgement of each assessor separately in a database. Relevance judgment is stored in a table with primary key as combination of the username of assessor, query and URL of the search result. When a query is fired, the input and output of each module is stored in an object. This object persists till expiry of session. The values captured in the object are displayed to the assessors on the tracker web interface. The assessor is allowed to modify the output of any module. The modified output is stored back in the object and used for overriding the output of the stages for which modification is done. The object stores change information till either session expires or user fires a different query. In the latter case, the object is used for storing information about new query. A separate table is created in database for maintaining revision history of the changes done for each query.

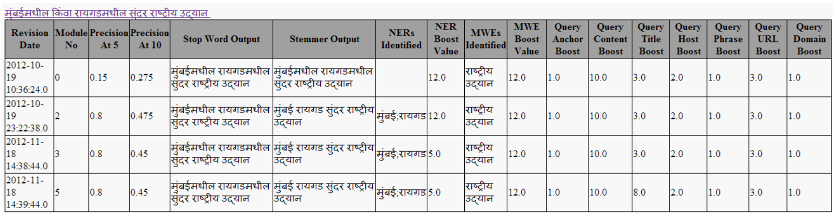
Figure 2 shows a screen of the tracker interface used for tracking output. The first column specifies the level in module hierarchy, second denotes module name and the last column shows the output of the corresponding output. Since it is pipeline architecture, the output of one module directly forms input of other module and hence inputs are not explicitly shown. The assessors are allowed to edit one level at a time which enables tracking incremental changes.



Level	Field Name	Field Value
0	Query	मुंबईमधील किंवा रायगडमधील सुंदर राष्ट्रीय उद्यान
1	StopWord Output :	मुंबईमधील रायगडमधील सुंद
2	Stemmer Output :	मुंबईमधील रायगडमधील सुंद
3	NERs identified (';' separated) :	
3	NER Boost :	12.0
4	MWEs identified (';' separated) :	राष्ट्रीय उद्यान
4	MWE Boost (Content) :	12.0
	Query Boost :	
5	Anchor Boost :	1.0
5	Content Boost :	10.0
5	Title Boost :	3.0
5	Host Boost :	2.0
5	Phrase Boost :	1.0
5	URL Boost :	3.0
5	Domain Boost :	1.0
	Lucene Query	
	+lang:mr domain:tourism +(MWE:"राष्ट्रीय उद्यान"^12.0 content:"राष्ट्रीय उद्यान"^12.0 title:"राष्ट्रीय उद्यान"^12.0) (url:मुंबईमधील^3.0 anchor:मुंबईमधील content:मुंबईमधील^10.0 title:मुंबईमधील^3.0 host:मुंबईमधील^2.0) (url:रायगडमधील^3.0 anchor:रायगडमधील content:रायगडमधील^10.0 title:रायगडमधील^3.0 host:रायगडमधील^2.0) (url:सुंदर^3.0 anchor:सुंदर content:सुंदर^10.0 title:सुंदर^3.0 host:सुंदर^2.0) (url:राष्ट्रीय^3.0 anchor:राष्ट्रीय content:राष्ट्रीय^10.0 title:राष्ट्रीय^3.0 host:राष्ट्रीय^2.0) (url:उद्यान^3.0 anchor:उद्यान content:उद्यान^10.0 title:उद्यान^3.0 host:उद्यान^2.0) url:"मुंबईमधील रायगडमधील सुंदर राष्ट्रीय उद्यान"~2147483647^3.0 anchor:"मुंबईमधील रायगडमधील सुंदर राष्ट्रीय उद्यान"~4 content:"मुंबईमधील रायगडमधील सुंदर राष्ट्रीय उद्यान"~2147483647^10.0 title:"मुंबईमधील रायगडमधील सुंदर राष्ट्रीय उद्यान"~2147483647^3.0 host:"मुंबईमधील रायगडमधील सुंदर राष्ट्रीय उद्यान"~2147483647^2.0	
	Editing Level (1-5)	*
	Precision At 5	0.15
	Precision At 10	0.275
	Submit	

FIGURE 2- Tracker Web Interface

Figure 3 shows how a revision history for each query is maintained. The boost values used for named entity, multi-word, title, content, *etc.* are part of system parameters which can be tuned using tracker. In this case change in stemmer, which is second module in the pipeline, causes significant rise in precision values. The third row in figure shows the effect of reducing named entity boost during query formulation. The fourth row depicts that change in title boost have no effect on precision values for this query.



मुंबईमधील किंवा रायगडमधील सुंदर राष्ट्रीय उद्यान

Revision Date	Module No	Precision At 5	Precision At 10	Stop Word Output	Stemmer Output	NERs Identified	NER Boost Value	MWEs Identified	MWE Boost Value	Query Anchor Boost	Query Content Boost	Query Title Boost	Query Host Boost	Query Phrase Boost	Query URL Boost	Query Domain Boost
2012-10-19 10:36:24.0	0	0.15	0.275	मुंबईमधील रायगडमधील सुंदर राष्ट्रीय उद्यान	मुंबईमधील रायगडमधील सुंदर राष्ट्रीय उद्यान		12.0	राष्ट्रीय उद्यान	12.0	1.0	10.0	3.0	2.0	1.0	3.0	1.0
2012-10-19 23:22:38.0	2	0.8	0.475	मुंबईमधील रायगडमधील सुंदर राष्ट्रीय उद्यान	मुंबई रायगड सुंदर राष्ट्रीय उद्यान	मुंबई;रायगड	12.0	राष्ट्रीय उद्यान	12.0	1.0	10.0	3.0	2.0	1.0	3.0	1.0
2012-11-18 14:38:44.0	3	0.8	0.45	मुंबईमधील रायगडमधील सुंदर राष्ट्रीय उद्यान	मुंबई रायगड सुंदर राष्ट्रीय उद्यान	मुंबई;रायगड	5.0	राष्ट्रीय उद्यान	12.0	1.0	10.0	3.0	2.0	1.0	3.0	1.0
2012-11-18 14:39:44.0	5	0.8	0.45	मुंबईमधील रायगडमधील सुंदर राष्ट्रीय उद्यान	मुंबई रायगड सुंदर राष्ट्रीय उद्यान	मुंबई;रायगड	5.0	राष्ट्रीय उद्यान	12.0	1.0	10.0	8.0	2.0	1.0	3.0	1.0

FIGURE 3- Revision History

6 User experiences of Tracker

Tracker was used by more than 100 developers and assessors all over India for tracing errors in multiple languages. A sample of overall feedback obtained is as follows:

- Positive points:
  - Useful tool to track query processing
  - Easy to evaluate and check the system's performance on varying boosts factors.
  - Revision History helps in comparing results.
- Improvements Suggested:
  - Should be extended to calculate precision values up to 25 results.
  - Should be extended to support dynamic resource updation while tracking changes.

Conclusion and perspectives

In this paper, we have highlighted the importance of having an error analysis tool like tracker to track individual modules of a large scale system. Tracker facilitates detection of errors in different modules of the search engine. Pseudo error correction of outputs helps discovering further errors in the system without making a change in the module. With an example of Marathi retrieval system, we have shown the use of tracker and its effectiveness in error analysis and analyzing performance of the system for various system parameters. Tracker is independent of the language of search and portable across search systems. This idea can be extended for making error analysis tools for any large scale system based on pipelined architecture.

Acknowledgments

Thanks to DAIICT, Gujarat for their contribution in developing a relevance judgement tool that is integrated with the tracker.

References

Text REtrieval Conference (TREC) Home Page (1992). *http://trec.nist.gov/*

The CLEF Initiative (Conference and Labs of the Evaluation Forum) – Homepage (2000). *http://www.clef-initiative.eu/*

NTCIR HOME (1999). *http://research.nii.ac.jp/ntcir/index-en.html*

FIRE - Forum for Information Retrieval Evaluation (2008). *http://www.isical.ac.in/~clia/*

Mugdha Bapat, Harshada Gune, Pushpak Bhattacharyya (2010). A Paradigm-Based Finite State Morphological Analyzer for Marathi. *Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WSSANLP)*, pages 26–34, the 23rd International Conference on Computational Linguistics (COLING), Beijing, August 2010**.**