King Julian: The N-Grams Based Language Model
Project Milestone-II

Himanshu S. Bhatt and Samarth Bharadwaj

March 13, 2012

1 Project Key Components

New Indexing strategy: Introduce a new option in Galago build, build-ngrams, with additional parameter n that indexes upto n-grams while also preserving tags.

- The current Galago indexing module computes Uni-grams of both stemmed and un-stemmed terms.
- The indexing mechanism also preserves the tags in extends.

New Retrieval algorithm: Based on [1], we use the probabilistic retrieval model for semi-structured data to formulate a structured query from an unstructured query. This is based on pre-occurrence of a given term in a certain zone in the corpus.
Query recommendation: The proposed retrieval system will suggest the recommended questions (query) for the information need based on the documents retrieved and will also suggest similar documents from the database.

2 Stack-overflow XML corpus

Description Stack-overflow is a creative common licensed Q&A website. The open corpus contains XML documents of question answers from each forum. As instructed, we are using the stackoverflow ($700MB$) and csetheory ($42.5MB$) forums.
Structure of Data

The data consists of several tags pertaining to the interface design on stackoverflow, such as user details, ratings, posts, etc. We consider the following tags obtained from this dataset.

Stackoverflow basically is structured around Question and Best answer as rated by the users. Additionally comments, multiple answers, revisions give important information around the subject of the question.

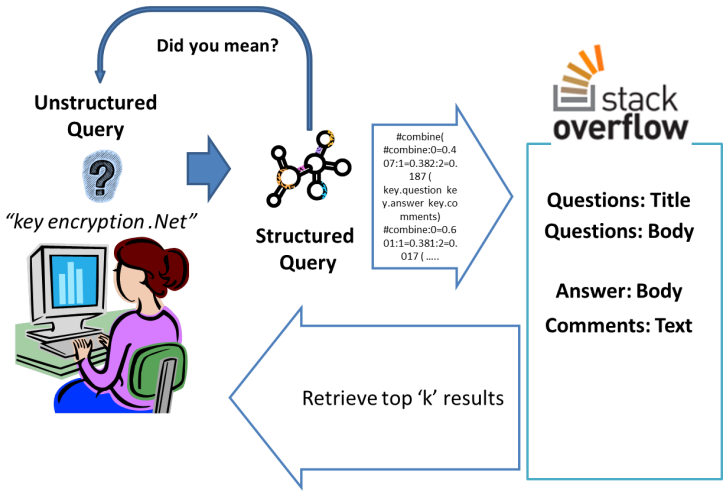

Figure 1: Example of face resurfacing that can change the appearance.

1. **Comments:** The information available on comments in not necessarily answer the question but provide some additional information
2. **Question:** The interface of stackoverflow tries to encourage content generators to provide a structured well phrased question, available as Title tag. The body contained details of the question and might include code.
3. **Answers:** For each question, the corresponding answers is a separate entry with Body and PostType tag. The answers are usually sorted based on the score of userfeedback. Limitation: The source code in the body section of questions and answers are not annotated seperately.

2.1 The structure of the database is summarized as follows

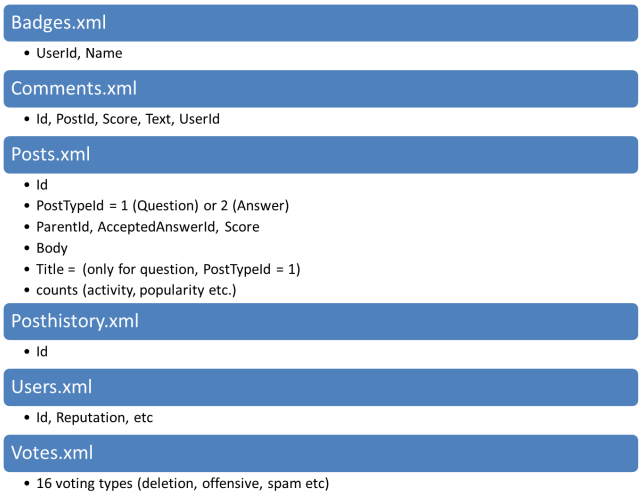

Figure 2: Illustrates the structure of the database.

2.2 Structure of Query

Our approach transforms a simple query to a structured query. Given a simple query like key encryption .Net against a movie collection composed of fields such as {question, answers, comments}, our goal is to transform it into the following structured query.

#combine(
#combine:0=0.407:1=0.382:2=0.187 (key.question key.answer key.comments)
#combine:0=0.601:1=0.381:2=0.017 (encryption.question encryption.answers encryption.comments)
#combine:0=0.927:1=0.070:2=0.002 (.Net.question .Net.answers .Net.comments))

In Galago, these transformations will be incorporated in the Traversal interface that defines how a given query can be transformed into another query.
Sample queries Example of keyword queries:

- Generalizing the FFT
- Space-bounded TMs and oracles

Examples of verbose queries:

- What is the theoretical basis of imperative programming?

- What are some career options for someone with a computer scientist master degree?

3 Evaluation Methodology

3.1 Evaluation metrics

- **Precision:** the fraction of retrieved documents those are relevant.
- **Mean Average Precision (MAP):** provides a single-figure measure of quality across recall levels provides good discrimination and stability. For a single information need, Average Precision is the average of the precision value obtained for the set of top k documents existing after each relevant document is retrieved, and this value is then averaged over information needs.

The retrieval evaluator object (provided by galagosearch.core.eval Class RetrievalEvaluator) computes a variety of standard information retrieval metrics commonly used in TREC, including geometric mean average precision (GMAP), mean average precision (MAP), and standard precision and recall. We will use this object for evaluation purpose.
We will also evaluate the effect of near-misses that are ignored by standard evaluation metrics. For this, each query will be evaluated across different n-gram models and we will report the performance. Evaluation will be performed for two types of queries:

1. Keyword query: Query comprises of keywords. Therefore, keyword queries are difficult to be modeled using using n gram models because the query terms do not generally follow the language flow and can be quite abrupt.
2. Verbose queries: Verbose queries comprises of full sentences. For retrieval using language models, verbose queries may be more effective and representative as they follow the language flow and can be predicted by the language models.

3.2 Our Approach

Calculating mapping probabilities: for weighing the models corresponding to each field/ context (such as questions, answers and comments) we predict the probability of each query term associated with a particular context. These mapping probabilities are used for weighing the scores generated from each filed models depending on the query terms. In our approach, probabilities for mapping each query term to a field in the document is estimated based on the Bayesian classification of query-term into one of field-level language models.
Let Q represent the query as $\{q_1, q_2, \cdot, q_m\}$, C be the collection of documents, and each document d has different fields represented as $\{f_1, f_2, \cdot, f_n\}$.
Posterior probability that q_i be mapped into f_j

$$P(E_j|q_i) = \frac{P_M(q_i|f_j)P_M(f_j)}{p(q_i)} \tag{1}$$

Package	Classes	Comments
Galagosearch.core.eval	Contains two evaluation methods 1) Single rank test 2) Rnak to rank comparision (statisiticals)	The methods compute serveral evaluation metrics for rank vs. judgement and rank vs. rank
Galagosearch.core.eval.stat	Statistical functions for evaluation	New metrics can be implemented here
Galago.core. index	The main indexing strategy is implemented in this package Interface Classes - IndexElement - QueryInterator Important Classes - PositionIndexReader - DocumentNameReader & DocumentLength	In order to perform n-gram indexing, the IndexElement is extended to multi-grams
Galagosearch.core.parse	Important Classes - Document ○ Contains document identifier, list of terms, plain text, Tags, metadata - TagTokenizer ○ Extracts text from tokens - Tags ○ Xml tags in document with attributes	The corresponding tag in the xml indicate question, answer zones
galagosearch.core.retrieval	Interface Classes The interface classes implement the classes from retrieval.structured and retrieval.query - Retrieval ○ The transformquery method converts the query into the complete representation	To implement PRM, a new query is constructed along with some methods to count terms per zone
galagosearch.core.retrieval. structured	Module processes the structured query Important Classes - Extent - ExtentDisjunctionIterator - ExtentInsideIterator ○ #inside operator is implemented. - FeatureFactory ○ Implements all Galago operators - Structured Retrieval	The classes for PRM in semi-structured doc is implemented using existing Galago operators
galagosearch.core.retrieval. query	Useful Classes - Node - SimpleQuery - StructuredQuery ○ Evaluates each node of structured queries by first evaluating operator and then traversing the query tree - Traversal ○ Interfaces the term by term iteration in query tree	
galagosearch.core.retrieval. traversal	Useful Classes - AddCombineTraversal - ImplicitFeatureTraversal	
galagosearch.core.scoring	Useful Classes - Implements several smoothing algorithms	

Figure 3: Illustrates packages of Galago and their functionality.

where $P_M(q_i|f_j)$: $\frac{tf(q_i,f_j)}{|f_j|}$ and $P_M(f_j)$: prior belief on element f_j.
The overall retrieval model is given as:

$$P(q|d) = \prod_{i=1}^{m} \sum_{j=1}^{n} P_M(f_j|q_i) P_{QL}(q_i|f_j)) \tag{2}$$

where $P_{QL}(q_i|e_j)$ is the element-level evidence.
For implementing the mapping probabilities (MP), we calculate the field-level collection frequency of each query-term, and the length of each field.

References

[1] J. Kim, X. Xue, and W. B. Croft. A probabilistic retrieval model for semistructured data. In *Proceedings of the European Conference on Advances in Information Retrieval*, pages 228–239, 2009.