An Intermediate Query Model for Structured Retrieval's Queries Construction

Keng Hoon Gan
School of Computer Sciences
Universiti Sains Malaysia
11800 USM Penang Malaysia
khgan@cs.usm.my

Keat Keong Phang
Faculty of Computer Science and Information Technology
University of Malaya
50603 Kuala Lumpur Malaysia
kkphang@um.edu.my

ABSTRACT

Looking at the amount of structured contents available on the web in recent years, we can be certain that the needs of structured retrieval systems are getting more prominent. In order to access the structured contents, information requests are formulated in the form of query languages supported by the structured retrieval systems. Nevertheless, information requests at this layer cannot be specified directly by end users. Hence, a common solution is to map information needs obtained from the application layer to the query language of a retrieval system automatically. Although rule-based approach can be used to generate query automatically, it is designed for a single target query language, whereas, there are not one but different types of structured queries with different forms of syntax used in a structured retrieval environment.

In this paper, we address the problem of constructing structured queries for structured retrieval by proposing an intermediate query model that can be used to construct different types of structured queries. This model allows flexibility of accommodating new query language by separating the semantic of query from its syntax. Firstly, rule-based approach is used to construct a generic form of query known as semantic query structure. Then, generation of query into target language is carried out by matching templates obtained from example-based method. Evaluations were carried out based on how well information needs are captured and transformed into a target query language. In summary, the proposed model is able to express information needs specified using query like NEXI. Xquery records a lower percentage due to its language complexity. We also achieve satisfactory query conversion rate with our example-based method, i.e. 86% (for NEXI IMDB topics) and 87% (NEXI Wiki topics) respectively compare to benchmark of 78% by Sumita & Iida in language translation.

iiWAS '14 December 04-06 2014, Hanoi, Vietnam

http://dx.doi.org/10.1145/2684200.2684286

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*query formulation, search process*

General Terms

Design, Theory

Keywords

Query Representation, Structured Retrieval, XML

1. INTRODUCTION

In information searching, structured retrieval allow users to access structured contents on the web. However, such access is normally masked from users via some interfaces. Information request from application interfaces are formulated in structured forms using methods like query languages (e.g. XQuery [2], NEXI [11]), forms (e.g. advance search [1, 12]) etc. whereby users can explicitly specify structures or markups in the query. Since the usage of these query languages is normally integrated as part of the retrieval system, they need to be constructed automatically.

Related works that automate the construction of structured queries fall in two major categories, i.e. templates or rules. In the first approach, [13] creates a set of grammar templates based on structured queries samples collected from INEX forum. Each grammar template corresponds to an individual information request. Similarly, [10] uses XSL Transformation to generate NEXI structured query from its generic query representation. As template-based approach may suffer from its coverage of structured query formats, hence, in the second approach, rules are defined to construct structured queries [7, 6]. E.g., in [7], a set of query construction operations are used to formulate target and constraint terms generated from keywords query into NEXI query language. In [6], XQuery clauses are generated using a set of procedures. Nevertheless, the limitation of rule-based method is that these operations or procedures are designed for particular structured query language. Thus, the ability to accommodate new structured query has become our concern. This motivates us to find a generic query construction process that can be adapted to new query language easily.

In this paper, we propose an intermediate query model that can be used to construct structured query in a more generic way. In general, construction of query in any structured query language involves a set of rules or operations [3].

However, as we intend to simplify the query construction process, rules (or operations) creation for each individual query language should be more flexible. We introduce an intermediate query model to replace individualized query construction process (e.g. QL1, QL2 in Figure 1) with a generic process (using Intermediate Query Model) that can be easily build using examples query. Information needs from search queries, web forms and other sources are captured in the form of a generic structure, called semantic query structure. In this structure, only query contents and its semantic are stored. The generation of query contents into a target language is carried out by finding suitable query template that has been captured (as syntax query structure) in the knowledge base.

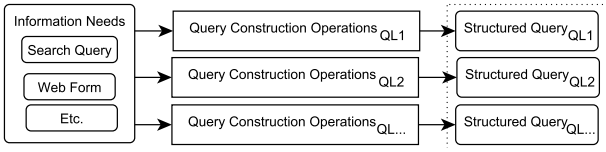


Figure 1: Individualized Query Constructions

2. A MODEL FOR STRUCTURED QUERY CONSTRUCTION

2.1 Preliminary

In this section, we shall introduce the schema which is the basic for designing the proposed intermediate query model. Two main features of a structured query are taken in account in schema definition i.e. query target and query constraint. As this model is designed for queries used for structured retrieval, it needs to at least be able to represent contents specified in popular queries like NEXI (e.g. //movie[about(.//director, "James Cameron") AND about(., best movie award)]' where query target is "movie" and query constraint is "director = James Cameron") and XQuery (e.g. for $c in document("geobase.xml")//city where $c/state = "Virginia" return <result> $c/text() </result> where query target is "city" and query constraint is "state = Virginia"). Here are two examples showing how query contents are represented in an intermediate query. First example shows how query contents from keywords query is represented.

EXAMPLE 1. *Query contents representation for keywords query "workshop by andrew trotman" written in xml schema language.*

```
<targetGroup>
    <target>
        <concept>workshop</concept>
    </target>
    <constraintGroup op="AND">
        <constraint>
            <concept>organizer</concept>
            <keyword>andrew trotman</keyword>
        </constraint>
    </constraintGroup>
</targetGroup>
```

Second example shows how query contents of an XQuery example is represented.

EXAMPLE 2. *Query contents representation for Xquery " for $c in document("geobase.xml")//city where $c/state = "Virginia" return <result> $c/text() </result>" written in xml schema language.*

```
<targetGroup>
    <target>
        <concept>city</concept>
    </target>
    <constraintGroup op="AND">
        <constraint>
            <concept>state</concept>
            <keyword>Virginia</keyword>
        </constraint>
    </constraintGroup>
</targetGroup>
```

From these examples, we proceed to present the components of intermediate query schema, such as the elements in an intermediate query, the attributes of elements in an intermediate query, the child elements, order and number of the child elements etc. Its schema definition and properties are described as follow.

2.2 Schema Definition

The definition of our proposed intermediate query schema is specified using XML schema definition language. Refer to Figure 2 for the illustration of the schema.

```
<xsd:element name="targetGroup" type=
"TargetGroupType"/>
<xsd:complexType name="TargetGroupType">
    <xsd:sequence>
    <xsd:element name="target" type="TargetType" max-
Occurs="unbounded"/>
    <xsd:element name="constraintGroup" type= "Constraint-
GroupType"/>
    <xsd:element name="targetGroup" type= "TargetGroup-
Type"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ConstraintGroupType">
    <xsd:sequence>
    <xsd:element name="constraint" type= "ConstraintType"
maxOccurs="unbounded"/>
    <xsd:element name="constraintGroup" type=
"ConstraintGroupType"/>
    </xsd:sequence>
    <xsd:attribute name="op" value="AND,OR"/>
</xsd:complexType>
<xsd:simpleType name="TargetType">
    <xsd:element name="concept" type="Concept"/>
```

Description of Schema.

The intermediate query schema consists of two main elements, i.e. target group element, TG, and constraint group element, CSG. A target group element, TG, has one or more target elements, T. Each target, T has one or more concept elements, C. More than one concept elements occurs if a concept is a path. Each structure of the path is represented as separate concept element under the same target. The concept of target is optional.

A target group can have one or more constraint groups, CSG. In general, common queries only have one target group. However there are cases where a specific target is required. In this case, additional target group can be attached

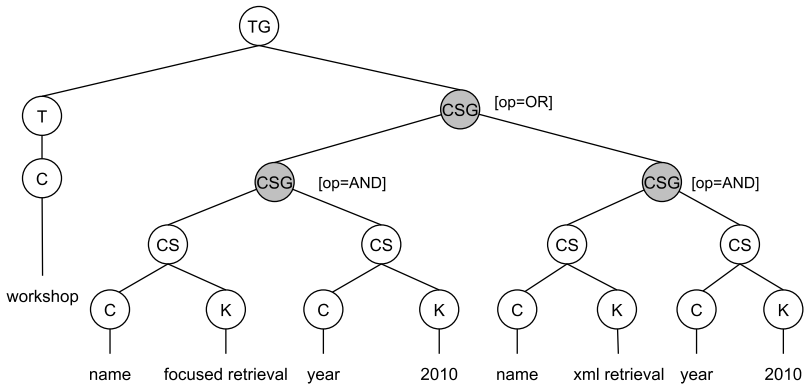
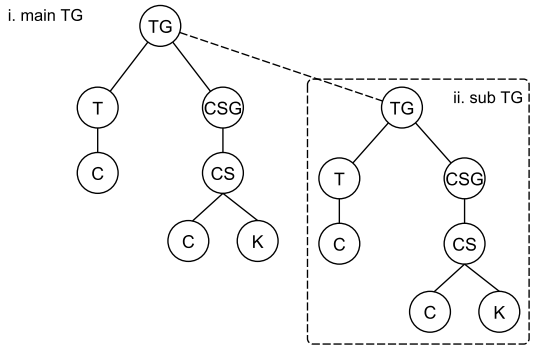


Figure 3: Representing Nested Information Needs



TG: target group CSG: constraint group
T: target CS: constraint C: concept K: keyword

Figure 2: Intermediate Query Schema

to existing target group as subtarget group as refined target elements (see ii. in Figure 2).

A constraint group element, CSG, has one or more constraint elements, CS. Each constraint, CS has a concept element, C, and a keyword element, K. The concept of constraint is optional. In the case when the concept is not specified, its value is null. A constraint group element has a logical operator as its attribute. The logical operator attribute can either have "OR" or "AND" as its value, to indicate where its constraint elements are disjunctive or conjunctive. Here, we show an example on how information needs with multiple constraint groups is represented using the schema (Figure 3).

2.3 Intermediate Query Model

In this section, we present a novel intermediate query model for query construction. This model consists of two sub structures, i.e. semantic query structure and syntax query structure. A schema matching function is used to find best match between these structures (Figure 4).

2.4 Semantic Query Structure

A semantic query structure captures the contents of query in generic manner so that it can be constructed into more than one target structured query later. Thus, it does not contain syntax of any structured query, but only contents (and logics). It captures three types of contents (see leaves in Figure 4), i.e. interpreted target concept, IT, interpreted constraint concept, IC, and interpreted constraint keyword, IK. These contents can be mapped to its syntax query structure counterpart to generate a structured query string.

One important feature that needs to be addressed by this semantic query structure is to bridge information needs from sources like keywords query, forms to structured forms. Hence, it should be able to support content representation for at least basic information needs for various structured queries. Examples of how simple information needs are represented are shown below.

Representing Single Content.

The simplest information needs for structured query comprises of one target and one constraint. Minimally, an intermediate query should be able to represent these two contents as they are the basic requirements of structured queries. For example, an information needs "I am searching for the workshop organized by andrew trotman" is interpreted into a target, "workshop" and constraint pair, with "organizer" as concept and "andrew trotman" as keyword. Both target and constraint are represented using a target group element as in Figure 5 a.

Representing Multiple Contents.

A simple information needs may not be limited to a single target or a single constraint; it can include few of them each, but they should be indicating a single intention. An example of information needs with multiple targets is shown in Figure 5 b. The needs, "I am searching for workshop or tutorial run by andrew trotman" specifies two targets. Although this query looks for two different targets, we understand that the intention is the same where this person is looking for information about events organized by andrew trotman.

A query can have multiple constraints. For example in Figure 5 c., the query is looking for the workshop venue organized by andrew trotman and jaap kamps. It has two constraints, "andrew trotman" and "jaap kamps". Both constraints are represented as a set of conjunctive constraints under a constraint group.

Representing Concept Path.

To represent a target with a path of multiple concepts, such as "/workshop/venue" in Figure 5 c., the concept path is split and represented as sibling concept nodes under the

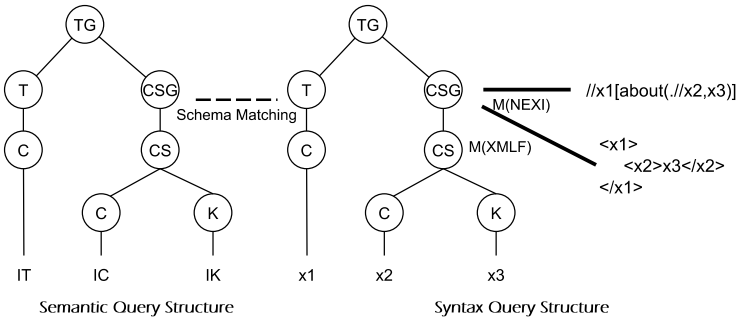


Figure 4: Intermediate Query Model

same target. Please note the difference of target representation between a. and c. in Figure 5. Although rarely concepts of target and constraint are specified as path, our schema allow such representation as it is supported by path-based structured query like NEXI.

2.5 Syntax Query Structure

Different from semantic query structure, a syntax query structure does not contain any query's contents. It only captures the template of structured query. The template of query is represented as two parts, a query structure and its corresponding query string. Variables between structure and string enable mappings of contents from the former to the latter. For the same query structure, there can be multiple mappings, with each mapping denotes a different type of query languages, e.g. *M(NEXI)* and *M(XML)* (see Figure 4). For each query language, X, there is a knowledge base.

Knowledge Base for Query Language, X.

A knowledge base, KB_X is used to store the query templates for a particular query language, X. As it is always hard to fill up an empty knowledge base from scratch, an example-based method is used to build a reasonable knowledge base with sufficient templates. Example-based method is commonly used in machine translation to build pairs of languages [9, 8]. For language translation, the knowledge base consists of pairs of source language and target language. Similarly, for our query language knowledge base, it consists of pairs of source format and target format. The source query is the structure form of the query example and the target query is the string form of the query example. As there may be repetitive examples, only unique pairs are stored. In Figure 6, we show an example on how a query template is generated from an example query. The template can be defined manually or can be automated generated with parser.

3. EVALUATION

In this section, the proposed intermediate query model is evaluated. Evaluation are carried out using structured queries from real query logs and test suites. The model is evaluated based on (i) the ability of its semantic query structure in capturing information needs, and (ii) the coverage of its syntax query structure (knowledge base) in supporting the construction of particular query language.

Metrics: Expressiveness and Coverage.

Two evaluation metrics are used. First, the model is evaluated by checking whether its semantic query structure can represent the contents of queries used for structured retrieval. The metric used is *expressiveness*, which is the fraction of the structured queries in a topic set that can be expressed using the proposed semantic query structure. To get an estimation of the expressiveness, we run the evaluation based on topic sets from both test suites or real query logs.

Each query, q, in a topic collection, Q, is scored as follows.

$$REP(q) = \begin{cases} 1 & \text{if } q \text{ can be represented as } q_{sem} \\ 0 & otherwise \end{cases}$$

, where q_{sem} is the semantic query structure of q.

The expressiveness can be summarized per topic collection by taking the average of query score for all the topics in the collection. Let us denote Q as the topics of a collection. The average over Q is

$$REP(Q) = \frac{1}{|Q|} \sum_{i=1}^{|Q|} REP(q_i)$$

We also measure its effectiveness in constructing multiple query languages. The metric used is *conversion success rate*, which is the fraction of test query that can be translated into query language, X, successfully, given a knowledge base of syntax query structure for language, X. Given a knowledge base of query language, X, each query, q, in a topic collection, Q, is scored as follows.

$$CVRT(q) = \begin{cases} 1 & \text{if } q_{sem} \text{ can be converted to } q_X \\ 0 & otherwise \end{cases}$$

, where q_{sem} is the semantic query structure of q and q_X is the structured query of q in language, X. The effectiveness can be summarized per topic collection by taking the average of query score for all the topics in the collection. Let us denote Q as the topics of a collection. The average over Q is

$$CVRT(Q) = \frac{1}{|Q|} \sum_{i=1}^{|Q|} CVRT(q_i),$$

3.1 Results and Discussions on Expressiveness

The model is first evaluated on whether it can represent the information needs expressed in structured queries. The test topics used are from either test suites or subset of query

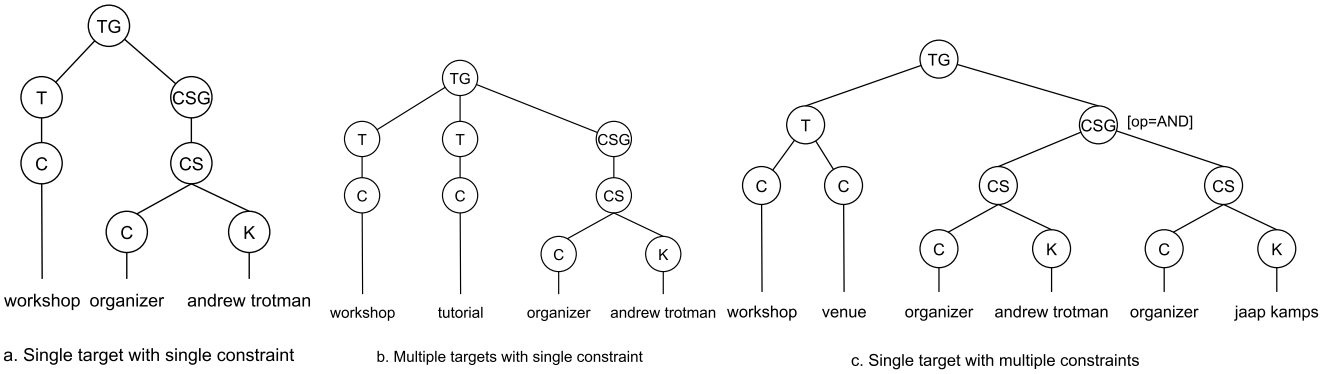


Figure 5: Representing Simple Information Needs

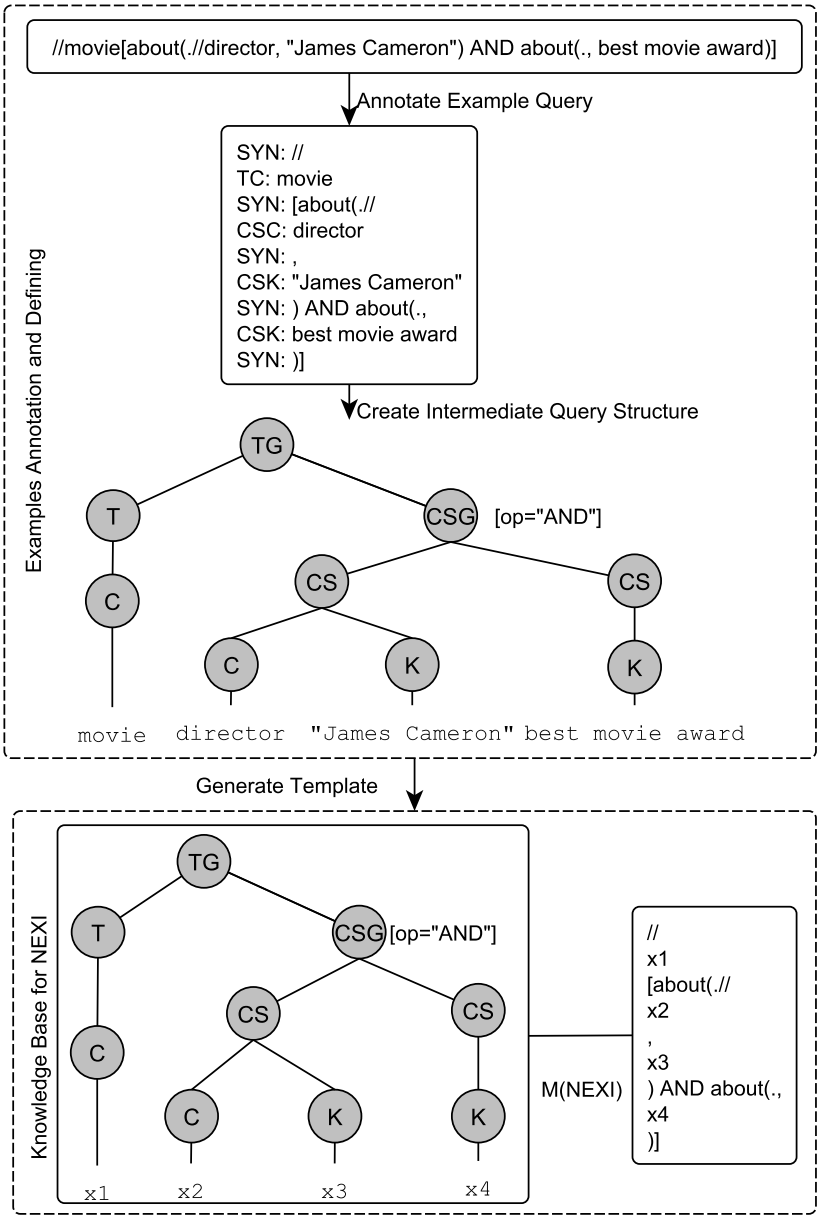


Figure 6: Knowledge Base Generation with Example Query

logs, e.g. topics from INEX IMDB and INEX Wiki are prepared by participants of the INEX forum. SIGIR Sites and DBLP topics are prepared in a similar manners. Geobase is a query logs created by real users of a publicly available web interface [4]. A summary of the result of expressiveness of the proposed query model is shown in Table 1.

Table 1: Expressiveness of Query Model

Name	Language	Total	REP (%)
INEX IMDB	NEXI	70	100%
INEX Wiki	NEXI	100	100%
Geobase	XQuery	90	60%
SIGIR Sites	Keywords	22	100%
DBLP	Keywords	20	100%

For INEX IMDB, INEX Wiki, SIGIR Sites and DBLP topics, we are able to express all the information needs specified in the topics. However for XQuery, only 60% can be fully represented with the semantic query structure of our proposed model. The % is lower as XQuery is a declarative language, where functional commands such as *let, count, max* etc. are oftenly used in topics. For these topics, our model can only represents partial information needs of the topics. E.g., when topic only contain commands like count, T_{51}: for $s in document("geobase.xml")//state where $s = "Rhode Island" return <result> {count($s/capital)} </result>. For this topic, we are able to represent the structure of "capital", but not the function to sum up the total elements of "capital".

Next, to show that the fairness of topics used. Their complexities are shown in Figure 7. Topic sets of two different INEX tracks, i.e. data centric track (INEX IMDB) and adhoc track (INEX Wiki) are used for this test. The distributions of topics show that our selected topics cover various complexities. This reassures the fairness of topics used in the evaluation of information needs representation.

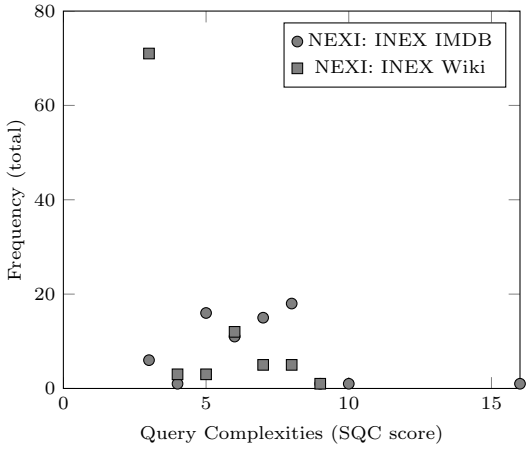


Figure 7: Query Complexities for Expressiveness Test

3.2 Results and Discussions on Coverage

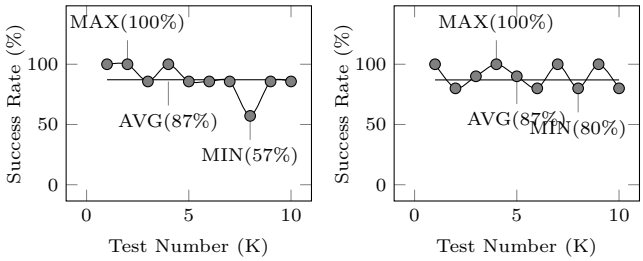
The proposed model is also evaluated on whether its idea of using knowledge base is able to support construction of query into a target language. The stability of knowledge base is tested using a cross-validation technique by [5]. This technique avoids biases of examples used in the knowledge base. The collection is partitioned into ten subsets. Nine sets are used as examples in knowledge base, and one set is used as test. This test was repeated 10 times, i.e. 10-fold cross-validation. This test is carried out using two NEXI topic collections. The performance of query conversion is shown in Table 2. The average success rate of query conversion is 86% for INEX IMDB and 87% for INEX Wiki collections. For both collections, we can see that the success rates for all the tests are scattered along the average line. An exceptional case of a rate of 57% is found in Figure 8a, which contain higher number of unconverted queries. This is probably due to the size of topic set which is too small, leading to uneven distribution of queries for training and testing. This phenomenon is not seen in Figure 8b where we use a bigger topic set.

Our evaluation method for this section is adopted from [9] in their work of example-based machine translation where they achieve an average of 78% success rate in using examples database for translation. It is expected that our average success rate is higher as we are dealing with queries transformation which are definitely simpler with smaller scope compare to natural language translation.

Table 2: Query Conversion Success Rate for NEXI Knowledge Base

Name	KB Size (No. of Eg.)	Test Size (No. of Query)	CVRT* (%)
INEX IMDB	63	7	86%
INEX Wiki	90	10	87%

*Average value is taken over 10 fold cross validation.



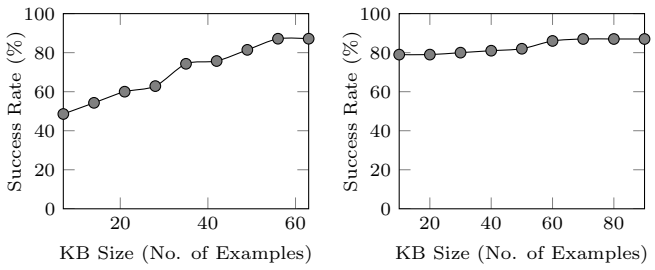
(a) NEXI: INEX IMDB (b) NEXI: INEX Wiki

Figure 8: The Success Rate of Query Construction using K-Fold Cross-Validation Technique

Further, we also test the stability of knowledge base by checking on the number of examples required for query construction. Figure 9a and Figure 9b show the relationship between the success rate of query construction and the number of examples. This graph shows that in general, for topics within the same collection, the more examples we have, the higher success rate of query transformation. The graphs stabilize when approximately 60 examples are used.

4. CONCLUSIONS

In this paper, we study the problem of structured queries constructions for multiple query languages in the field of structured retrieval. An intermediate query model is proposed. The results of our evaluations show the potential usage of the proposed model in constructing structured queries



(a) NEXI: INEX IMDB (b) NEXI: INEX Wiki

Figure 9: The Success Rate of Query Construction over KB Size

(e.g. NEXI, XQuery) by using example-based method. We have also shown that the model can be used for more than one type of structured query languages.

For future directions, we shall explore on extending the current query model to accommodate more types of mappings such as between the source queries like questions and target queries like SQL queries.

5. ACKNOWLEDGEMENTS

The authors gratefully acknowledge Universiti Sains Malaysia and the School of Computer Sciences, Universiti Sains Malaysia for their contributions in fundings assistance.

6. REFERENCES

[1] C. D. Barranco, J. R. Campaña, and J. M. Medina. Towards a xml fuzzy structured query language. In *EUSFLAT Conf.*, pages 1188–1193, 2005.

[2] D. D. Chamberlin. Xquery: An xml query language. *IBM Systems Journal*, 41(4):597–615, 2002.

[3] K. H. Gan and K. K. Phang. A query transformation framework for automated structured query construction in structured retrieval environment. *J. Information Science*, 40(2):249–263, 2014.

[4] M. Jayapandian and H. V. Jagadish. Automated creation of a forms-based database query interface. *PVLDB*, pages 695–709, 2008.

[5] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[6] J. Li, C. Liu, R. Zhou, and B. Ning. Processing xml keyword search by constructing effective structured queries. In Q. Li, L. Feng, J. Pei, X. S. Wang, X. Zhou, and Q.-M. Zhu, editors, *APWeb/WAIM*, volume 5446 of *Lecture Notes in Computer Science*, pages 88–99. Springer, 2009.

[7] D. Petkova, W. B. Croft, and Y. Diao. Refining keyword queries for xml retrieval by combining content and structure. In *ECIR*, pages 662–669, 2009.

[8] H. L. Somers. Review article: Example-based machine translation. *Machine Translation*, 14(2):113–157, 1999.

[9] E. Sumita and H. Iida. Experiments and prospects of example-based machine translation. In D. E. Appelt, editor, *ACL*, pages 185–192. ACL, 1991.

[10] X. Tannier. From natural language to nexi, an interface for inex 2005 queries. In *INEX*, pages 373–387, 2005.

[11] A. Trotman and B. Sigurbjörnsson. Narrowed extended xpath i (nexi). In *INEX*, pages 16–40, 2004.

[12] R. van Zwol, J. Baas, H. van Oostendorp, and F. Wiering. Bricks: The building blocks to tackle query formulation in structured document retrieval. In *ECIR*, pages 314–325, 2006.

[13] A. Woodley and S. Geva. Nlpx at inex 2006. In N. Fuhr, M. Lalmas, and A. Trotman, editors, *INEX*, volume 4518 of *Lecture Notes in Computer Science*, pages 302–311. Springer, 2006.