"Did You Know?" A Rule-based Approach to Finding Similar Questions on Online Health Forums

Jianglei Han
SAP Research & Innovation
#14, CREATE Tower, 1 Create Way
Singapore 138602
ray.han@sap.com

Naveen Nandan
SAP Research & Innovation
#14, CREATE Tower, 1 Create Way
Singapore 138602
naveen.nandan@sap.com

Aixin Sun
School of Computer Engineering
Nanyang Technological University
Singapore 639798
axsun@ntu.edu.sg

***Abstract*—This paper describes our system submitted for the ICHI 2015 Healthcare Data Analytics Challenge. Given a relatively large corpus of questions posted by users on online health forums, for a newly posted question (i.e., query question), our task is to find three most similar questions from the corpus. Our system employs *Elasticsearch*, a search server based on *Lucene*, at its core. The corpus of existing questions is indexed with n-grams. To search for most similar questions, the query question is re-written to a keyword-based query based on rules by considering multiple text components including title, key phrases, and noun phrases extracted from the question content.**

I. Introduction

Online social health forums are emerging as most important sources of medical information for Internet users. Compared to the traditional content-centric websites, where information is often static and structured, social forums provide a dynamic and interactive platform for users to ask and answer questions. Particularly in healthcare domain, questions asked on online forums are much longer than keyword-queries issued to search engines, allowing users to describe their conditions in much more details. More importantly, questions about similar medical conditions may have been asked and answered by other users in the forum. Effectively finding questions for most similar medical conditions enables users receiving immediate advices from existing discussions without further waiting.

The ICHI 2015 Healthcare Data Analytics Challenge provides a sample question corpus of 95 questions. Each question has a unique ID, a title that summarizes the question, and text content that details the question. Figure 1 gives an example question. The length of the questions ranges from 11 to 216 words. An additional set of 10 query questions is provided in the same format for development and evaluation purposes. As a challenge, the dataset does not provide groundtruth labels.

Title:	Metamorfin Advice Please
Content:	I am just wondering if anyone knows if you take more than one pill of your usual dose of Metamorfin can it hurt/affect you in any way? I had my tea but I couldn't remember if I had my Metamorfin afterwards. Should I take a pill if I'm not sure. or should I just leave it?

Fig. 1. A sample question with title and content (ID: 101)

The challenge in retrieving similar questions lies in the different ways each user may use to describe similar problems and/or symptoms. It has also been identified that the terms used by general users and professionals for the same symptom varies [1]. Through this challenge, we take a rule-based approach that mainly focuses on keyword extraction and query construction to build a system that retrieves most similar medical questions to existing ones.

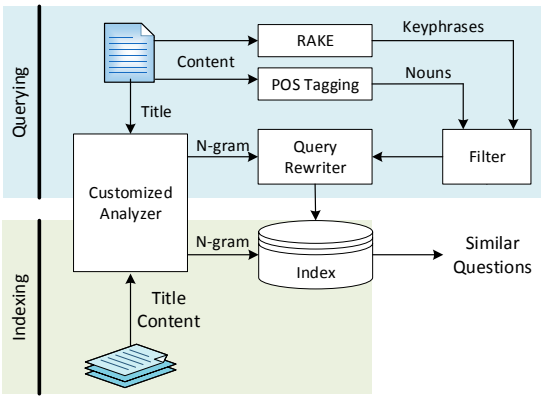

Fig. 2. System framework (best viewed in color)

II. System Design and Implementation

Our system consists of two main components for indexing and querying respectively, shown in Figure 2. The question corpus is indexed as a collection of JSON documents. The main focus of the system is the querying component, or more specifically, the extraction of keywords or keyphrases from a given query question, shown in the upper portion of Figure 2.

The system is built in Python 2.7 with open-source tools/-packages including *Elasticsearch* for indexing and searching, *RAKE* (Rapid Automatic Keyword Extraction)[1] for extracting keyphrases, and *NLTK* (Natural Language Toolkit)[2] for part of speech (POS) tagging. Next, we detail the components.

A. Indexing

The question corpus is indexed with *Elasticsearch*, as a collection of JSON documents. The indexing process involves tokenization, lower-casing, and stemming. Existing studies show importance of n-grams in processing short text [2]. Because questions are usually short, we choose to index word

[1] https://pypi.python.org/pypi/python-rake/1.0.5
[2] http://www.nltk.org/

```
"cu_analyzer": {
  "type": "custom",
    "tokenizer": "standard",
    "filter": [
      "lowercase",
      "kstem",
      "shingle_filter"
    ]
}
```

Fig. 3. Customized analyzer used for indexing

n-grams (also known as shingles in Elasticsearch), where $1 \leq n \leq 4$. Note that, $n = 1$ means individual words. Figure 3 details the customized analyzer used in our indexing, where `kstem` stems a word to a base form which is also a valid word in the dictionary. We observe that using kstem improves true positive hits based on manual evaluation of the results.

B. Query Rewriting

Given a query question, the key challenge is extracting keywords from the question for searching the index, i.e., query formulation. The selected words should cover the key idea of the query question, yet too many keywords are likely to result in a large number of false positives. Illustrated in Figure 2, we consider the title of a query question as a summary of a question which usually covers the main topic of the question. Hence, n-grams in title are considered as part of the query (recall that we index the question corpus with n-grams). For the content part of the query question, we extract keyphrases and nouns based on the assumption that such words contain more important information than others. For this purpose, we employ RAKE to extract the keyphrases (including individual keywords) and use NLTK to obtain the nouns.

RAKE extracts a list of candidate keywords. The list consists of n-grams up to four words ($1 \leq n \leq 4$), and each of which is assigned an importance score. From this list, we first select the words/phrases which appear in the title of the query question to be query keywords. Next, we select the words/phrases that are assigned the highest scores by RAKE. Note that, there could be multiple words/phrases sharing the same highest score. Among the words/phrases with the highest score, the phrases with more than 1 word are kept as part of the query keywords, and the unigrams which are nouns are also used as query keywords. The nouns are based on POS tagging by NLTK.

Queries in *Elasticsearch* are in the format of *Query DSL*, defined in JSON. Shown in Figure 4, a document should match at least one of the two `"multi_match"` query clauses in the `should` block. In the first query clause, the query keywords are the n-grams obtained from the title of the query question (through customized analyzer). The default `OR` operation is applied to the multiple n-grams. The `multi_match` indicates that matching are in both *title* and *content* fields. However, because the title is considered more important than content, the `boosting` feature is used to reward documents which contain matching words in the title [3]. In this case, the boosting for title is set to 1.5 empirically in the relevance score computation for a matching document. The `most_fields`

```
"query": {"bool": {
  "should": [
      { "multi_match" : {
        "query": title,
        "type": "most_fields",
        "fields":["title^1.5", "content"],
        "analyzer": "cu_analyzer" }
      },
      { "multi_match" : {
        "query": " ".join(high_key),
        "type": "cross_fields",
        "fields": ["title^10", "content"]}
      } ]
}   }
```

Fig. 4. Rule-based boolean queries

option indicates that the retrieved documents should have as many words as possible in the same field, and the relevance score should come from the best-matching field. In the second query clause, the query keywords are the keyphrases and nouns obtained from RAKE and NLTK after filtering. The `cross_fields` is used by the keywords in the query to match as many terms as possible in both fields and the relevance score is computed by combining the scores from both fields. The boosting for title is set to 10.

The relevance scores of the matching fields are computed by *Elasticsearch* with its default similarity function which is based on the TF-IDF feature vector representation of the fields in documents.[3] The final score of a matching document is the sum of the scores from the two query clauses.

III. Conclusion

In this paper, we describe a rule-based question retrieval system that is capable of searching for similar questions. The index and query logic are optimized for the best possible empirical performance based on manual observation. The method shows some extent of effectiveness in finding questions with keywords that match with the title or content. However, our manual evaluation could be subjective as compared to experts' opinions.

Acknowledgement: We would like to thank the Economic Development Board and the National Research Foundation of Singapore for partially funding this research. This work was also partially supported by Singapore MOE AcRF Tier-1 Grant RG142/14.

References

[1] Q. T. Zeng, T. Tse, G. Divita, A. Keselman, J. Crowell, and A. C. Browne, "Exploring lexical forms: First-generation consumer health vocabularies," in *AMIA Annual Symposium Proceedings*, 2006, p. 1155.

[2] X. Wang, A. McCallum, and X. Wei, "Topical n-grams: Phrase and topic discovery, with an application to information retrieval," in *Proc. IEEE ICDM*, 2007, pp. 697–702.

[3] W. W. Cohen and Y. Singer, "Context-sensitive learning methods for text categorization," in *Proc. SIGIR*, 1996, pp. 307–315.

[3] https://www.elastic.co/guide/en/elasticsearch/guide/current/scoring-theory.html