AskDragon: A Redundancy-Based Factoid Question Answering System with Lightweight Local Context Analysis

Xiaohua Zhou[1], Palakorn Achananuparp[1], E. K. Park[2], Xiaohua Hu[1], and Xiaodan Zhang[1]
[1]College of Information Science & Technology, Drexel University, Philadelphia, PA 19104, USA
[2]School of Computing and Engineering, University of Missouri at Kansas City, Kansas City, MO 64110, USA

Abstract
We introduce our QA system *AskDragon* which employs a novel lightweight local context analysis technique to handling two broad classes of factoid questions, *entity* and *numeric* questions. The local context analysis module dramatically improves the efficiency of QA systems without sacrificing high accuracy performance.

Categories and Subject Descriptors
H.3.4 [**Systems and Software**]: Question-answering (fact retrieval) systems

General Terms
Algorithm, Experimentation, Performance

Keywords
Question Answering, Redundancy-Based Approach, Local Context Analysis, Answer Generation, Answer Scoring

1. System Architecture
Our factoid question answering system, AskDragon, consists of nine components as shown in Figure 1. The main contribution of our system lies in the local context analysis module which corresponds to two components, *candidate generation* and *candidate scoring*. Many redundancy-based QA have to weigh answers based on the exactness of the query from which the answer is extracted. Candidates from exact query receive a higher score while those from less strict or inexact query get lower score. This scoring method is still too coarse. In addition, it is extremely expensive to submit multiple queries at different exactness to the search engines. To overcome the problems, the local context analysis method attempts to capture syntactic relations between candidate answers and natural language questions in order to find the correct answers. To keep the system lightweight, we do not utilize any deep natural language processing.

AskDragon submits only one *less strict* query or *inexact* query per question to Google search engine. Given the retrieved set of documents from Google, AskDragon then selects a small subset of high-quality sentences to perform further computational process such as part-of-speech tagging and pattern-based answer generation. With this method, the efficiency of the QA system is considerably improved over traditional QA systems even though the local context analysis itself requires some additional computation.

Candidate Answer Generation: AskDragon uses pattern matching rules to generate candidate answers. Two sets of rules are developed to generate numbers and entities, respectively. The extraction of numeric candidates from a sentence to find the maximal sequence of tokens satisfying the following four conditions: (1) begins with a number, (2) ends with a number or a unit, (3) no two concatenating symbols in a row, and (4) any token must be a number, a unit, or a concatenating symbol. In a case of entity generation, we focus on finding noun phrases. To handle irregular entities such as movie and song titles which may contain punctuations and verbs, we further perform candidate expansion to recover the irregularities.

Candidate Answer Scoring: Three factors are defined for scoring candidate answer. The first factor is the similarity of a sentence to a query, the second factor is the distance of the candidate answer to non-functional query words in a sentence, and the third factor is semantic compatibility of the candidate answer with the question. The three scoring factors make it possible to assign reasonable scores to different candidates even if they are generated from the sentences returned by the same query

The evaluation on common test question sets shows that the performance of AskDragon is as good as or even higher than that of state-of-the-art QA systems such as Aranea and QUANTA while running much faster than its counterparts.

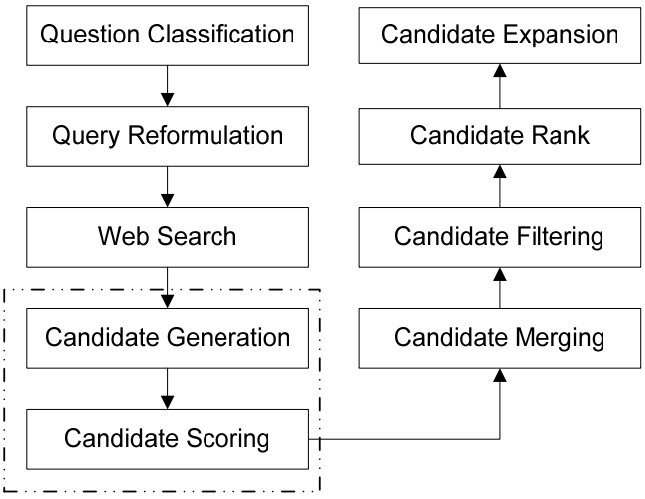

Figure 1. The architecture of AskDragon QA system.

JCDL'09, June 15–19, 2009, Austin, Texas, USA.
ACM 978-1-60558-322-8/09/06.