A* Search Control in a Swift-Based Closed-Domain Question Answering System

Abstract

In this paper, the effects of an A* search control algorithm are examined on a small-knowledge base closed-domain question answering program in Swift. It is hypothesized that compared to no search control, a search control using A* will increase the speed at which the program finds an answer, and that the answer will be at least as accurate on average. Experiments to test this hypothesis were conducted using three different knowledge bases, and the results of the experiments reject the hypothesis.

1 Introduction

There has been tremendous recent interest in question answering systems that are capable of learning from information provided by users and responding to questions with relevant answers.[6] These question answering systems are often part of "chatterbots", programs that can communicate with individuals using natural language. Some popular chatterbots, such as Siri and Cortana, have a large body of information that is modified and updated manually. Other programs, such as Microsoft's new bot Tay, build up information from user input. While very accurate, programs like these rely on remote services to function. Without access to an Internet connect, Siri and Cortana become useless. Thus, there exists a use for question answering problems that return answers using a locally stored knowledge base. This paper examines the use of a basic closed-domain question answering program written in Swift.

Although there is a number of open source question answering applications available online, none exist in Swift, a relatively very new open-source programming language created by Apple. I created a simple question answering program in Swift that retrieves paragraphs of information from a local file and attempts to answer questions based on this local information. This program was designed to be a closed-domain question answering system instead of open-domain, which means that it deals with a knowledge base and corresponding questions for one subject at a time, such as a country, field of study, or event.

The program is capable of returning an answer in one of two ways. The first is without any form of search control, and the second is with an A* algorithm search control. Furthermore, in addition to being closed-domain, the program is designed to hold a knowledge base from paragraphs of text no larger than 10 kilobytes, about the size of an medium-sized Wikipedia article.

The experiments of this paper involves testing the speed and accuracy of answers returned for simple and difficult questions using no search control and using A* search control. A similar

experiment was conducted in the paper by Tatsunori Mori, et al. mentioned in the literature review [8]. In this paper, a domain-ambiguous question answering program was create with the ability to return answers from a local knowledge base of ambiguous size using three options: no search control, A* search control with maximum scores only, and A* search control with approximate and maximum scores.

It is hypothesized that this Swift program will return an answer at least twice as quickly using the A* algorithm search control compared to no search control, and that the ratio of correct answers to total answers will be at least as high compared to no search control.

2 Literature Review

2.1 Question Answering Basics

There are a number of papers that explore the basics of question answering. One of these is "Memory, Knowledge, and the Answering of Questions" by Donald A. Norman [9]. Cited by a number of papers in this literature review, it explores the concept of asking questions separated from its technical applications. It puts forth the argument that the proper answering of questions is comprised of more than just information retrieval. Rather, a question answering agent must have a knowledge of the questioner, the question, and the world for a proper answer. The paper also puts forth a formal structure for representing information and gives examples. "Strategy selection in question answering" by Lynne M. Reder [11] builds off the Norman paper by exploring its proposed framework more in-depth, with experiments the focus on its validity.

"A* Search Algorithm for Question Answering" [8] by Tatsunori Mori, et al. states that typical question answering systems accept questions of five different categorical types: who, when, where, what and how.

2.2 Question Answering Techniques

Some papers go into more depth about question answering techniques, such as "Learning Surface Text Patterns for a Question Answering System" by Deepak Ravichandran and Eduard Hovy [10], which explores techniques for acquiring surface text patterns automatically from large bodies of text, and how they can be applied specifically for answering open-domain questions.

Similarily, "The Problem of Precision in Restricted-Domain Question-Answering. Some Proposed Methods of Improvement" by Doan-Nguyen Hai and Leila Kosseim [5], proposes a number of suggestions to improve the precision of answers given by closed-domain QA systems that fall under two categories: improvements to the information retrieval module of a QA system, and improvements to the final results.

"Learning Surface Text Patterns for a Question Answering System" by Deepak Ravichandran and Eduard Hovy [10] explores techniques for acquiring surface text patterns automatically from large bodies of text, and how they can be applied specifically for answering open-domain questions. The paper "Data-Intensive Question Answering" [3] focuses on developing a QA system that takes advantage of vast quantities of text found online in real-time. It discusses the possibility of searching the Web for answers to a question, collecting strings of relevant text, and producing the most likely answer.

"Exploiting Redundancy in Question Answering" by Charles L. A. Clarke et al. [4] seeks to answer the question of how to find potential answers to questions from large bodies of data, and

then assess the accuracy of each potential answer out of multiple answers. It proposes breaking up this question into two processes: the first process is to retrieve data that could be an answer by arbitrary passage retrievals that match keywords, and the second process is to search for redundant answers in order to reach the most accurate final answer.

Finally, the paper by Tatsunori Mori, et al. mentioned in the previous subsection explores basic methods of determining an answer for questions posed to a closed-domain question answering system with a locally stored knowledge base, and states that fusions of Information Retrieval (IR) and Information Extraction (IE) are typical methods. It explores implementing an A* Search algorithm within a question answer system with a large local knowledge base, and finds that it it increases the speed of finding an answer fourfold, but that the accuracy of the answer is decreased.

2.3 Papers on Chatterbots and Similar Applications

A number of other question answer papers focus on specific implementations in the form of chatterbots. "Linguistic Knowledge and Question Answering" by Gosse Bouma [1] provides an example of a question answer system named Joost. It goes into some detail about how Joost acquires lexical knowledge for both open and closed-domain question answering.

"An Analysis of the AskMSR Question-Answering System" [2] looks at a more complex system that has greater dependency on data redundancy. The paper also explores strategies for predicting when the system is likely to give the wrong answer to a question, which can be valuable information.

Lastly, the paper "Scaling Question Answering to the Web" by Cody Knok, et al. [7] introduces Mulder, an open-domain QA system that was the first to be available on the Web. It goes into detail about the architecture of the system, and compares the efficiency of the system with using a standard search engine like Google to answer questions.

3 Approaches to Implementation and Search

The program was written in Swift as an Xcode Playground program. Although typical question answering systems accept questions of the categorical types who, when, where, what and how, only "where" and "who" were implemented for this program for simplicity.

It employs two different methods of determining the best answer to a posited question.

3.1 No Search Control

The first method is without search control. In this method, for each sentence of the knowledge base, the similarity between the sentence and the question is calculated, and saved as a score for the sentence. The similarity is determined by two things. First, by counting the number of keywords which appear in both the sentence and the question. And second, by matching the sentence type with the question type. For example, if the question contains "where", a sentence with words such as "in" and "on" will have a higher score.

The sentence with the highest score is considered the answer, and the answer given to the user is extracted from this sentence.

3.2 A* Search Control

To reduce the amount of necessary processing involved in evaluating each sentence for the answer, the A* search algorithm was implemented as a second search option.

Here, instead of evaluating each sentence, only sentences with at least one matching word will be evaluated for a score. So if the question is "Who was the first country in Asia to host the Summer and Winter Olympic Games?", a sentence such as "Japan was the first Olympics country" would be evaluated, while a sentence like "Japan hosts sporting events" would not be evaluated. Simple words like "it" are excluded from this.

The sentence with the highest score among these sentences is considered the answer.

4 Experimental Design & Results

4.1 Knowledge Base

Because the Swift program is closed-domain, the experiment included inputting paragraphs of one subject at a time. Three subjects were used: "Japan," "The Beatles", and "Coffee." The paragraphs were obtained from Wikipedia articles. The paragraphs for each subject totaled no more than 10 kilobytes. The following example is an excerpt from one of the paragraphs:

> From the 12th century until 1868, Japan was ruled by successive feudal military shoguns who ruled in the name of the Emperor. Japan entered into a long period of isolation in the early 17th century, which was ended in 1853 when a United States fleet pressured Japan to open to the West.

4.2 Posited Questions

For each of the subjects, five questions were tested. These questions are all questions that can be answered from the knowledge base by a human. For simplicity, all questions were "where" and "who" questions, such as "Where is Japan?" and "Who was the first country in Asia to host the Summer and Winter Olympic Games?"

4.3 Results

Each question was run three times using Xcode on the same 2015 MacBook. A timer built into the program was used to track the time it took to generate the answer to each question. The times for search control type by subject is shown in Table 1.

Table 1: Speed of answer generation (seconds)

	No Search Control	A* Search Control
Subject 1	0.0025	0.0113
Subject 2	0.0018	0.0037
Subject 3	0.0038	0.0070
Average	0.0027	0.0073

In addition to time, the percentage of correct answers were recorded. Each answer was determined to be correct or incorrect manually. The correctness of the answers for each control type by subject is shown in Table 2.

Table 2: Percentage of answers that were correct

	No Search Control	A* Search Control
Subject 1	20%	0%
Subject 2	40%	20%
Subject 3	20%	20%
Average	26.7%	13.3%

5 Analysis of Results

Table 1 of the results show that the usage of the A* search algorithm to find an answer actually slowed down the execution of the time it took to find an answer. The program was 2.7 times faster when it was using no search control than when it was using A* search control, even though it was hypothesized that it would take more time.

Although A* search took longer, according to Table 2, A* provided less accurate answers. While no search control resulted in answers that were correct 26.7% of the time, A* search control resulted in answers that were correct only 13.3%, two times less.

6 Conclusion

The result of this experiment contradict the hypothesis that the program will return an answer at least twice as quickly using the A* algorithm search control compared to no search control, and that the ratio of correct answers to total answers will be at least as high compared to no search control. This could mean that the A* search algorithm would be beneficial only for larger knowledge bases, such as the ones in the experiments in the paper by Mori, et al. [8].

The results may also be the result of a poor implementation of the A* algorithm, or another portion of the program. In the future, this algorithm should be tested on more subject areas with knowledge bases of about the same size, using questions that cover all five main question answer types: who, when, where, what and how.

References

[1] G. Bouma. Linguistic knowledge and question answering. In *Proceedings of the Workshop KRAQ'06 on Knowledge and Reasoning for Language Processing*, pages 2–3. Association for Computational Linguistics, 2006.

[2] E. Brill, S. Dumais, and M. Banko. An analysis of the askmsr question-answering system. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 257–264. Association for Computational Linguistics, 2002.

[3] E. Brill, J. J. Lin, M. Banko, S. T. Dumais, A. Y. Ng, et al. Data-intensive question answering. In *TREC*, volume 56, page 90, 2001.

[4] C. L. Clarke, G. V. Cormack, and T. R. Lynam. Exploiting redundancy in question answering. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 358–365. ACM, 2001.

[5] H. Doan-Nguyen and L. Kosseim. The problem of precision in restricted-domain question-answering. some proposed methods of improvement. In *Workshop on Question Answering in Restricted Domains. 42nd Annual Meeting of the Association for Computational Linguistics (ACL-2004)*, pages 8–15, 2004.

[6] Y. Kodratoff and R. S. Michalski. *Machine learning: an artificial intelligence approach*, volume 3. Morgan Kaufmann, 2014.

[7] C. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the web. *ACM Transactions on Information Systems (TOIS)*, 19(3):242–262, 2001.

[8] T. Mori, T. Ohta, K. Fujihata, and R. Kumon. A* search algorithm for question answering. In *NTCIR*, 2002.

[9] D. A. Norman. Memory, knowledge, and the answering of questions. 1972.

[10] D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 41–47. Association for Computational Linguistics, 2002.

[11] L. M. Reder. Strategy selection in question answering. *Cognitive psychology*, 19(1):90–138, 1987.