A FACTOID QUESTION AND ANSWER SYSTEM

Dhruv Sangvikar

Department of Computer Science, University of Texas at Dallas

dgs160230@utdallas.edu

***Abstract*— A factoid Question and Answer system which understands the question given in natural language and outputs answer using Wikipedia as the backend database. The type of question and the named entities would be extracted from the query, answer retrieved from Wikipedia and content from Wikipedia would be parsed for the right answer based on the question type. The domain of the system would be restricted to movies.**

***Keywords*— Question and Answer System, Information extraction, Named entity recognition, Wikipedia QnA**

I. Introduction

Open-domain question answering and story comprehension have become important directions in natural language processing. Question answering is a retrieval task more challenging than common search engine tasks because its purpose is to find an accurate and concise answer to a question rather than a relevant document. The work presented here is the result of an attempt to develop a practical system which can understand natural language questions in the movie domain, search for the answers on wikipedia and extract the appropriate answer from the content returned. The approach consists of using NLTK tokenization, NLTK POS tagging, chunking, chunking grammar rules, information extraction using Regex. The user interface is provided using a simple web application to enhance and ease the usability of the application.

The document consists of 4 sections namely "Related work", "Proposed System", "Implementation", and "Resources used".

II. Related Work

There has been a lot of good work to formalize the approach taken for a question and answer system. I have taken a lot of help from the work by *Xin Li and Dan Roth* for "Learning Question Classifiers". Their approach consists of developing a hierarchical classifier that is guided by a layered semantic hierarchy of answer types and is able to classify questions into fine-grained classes. The overall flow of the system was based on the approach given in the Jurafsky's chapter on Question Answering. These are the main sources on which this work is based on.

III. Proposed System

This system follows the approach specified by Jurafsky which consists of Question Processing, Content Retrieval, and Answer Extraction. The following diagram details these processes.

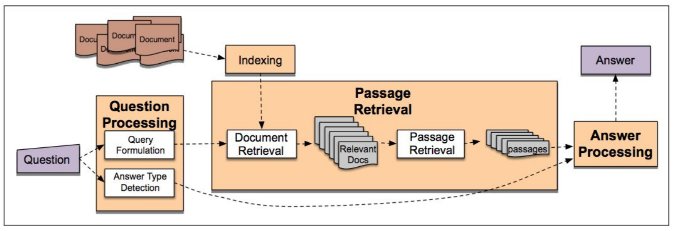


Fig. 1 IR-based factoid question answering has three stages: question processing, passage retrieval, and answer processing

Here the Question is provided by the user, which is processed in the system and the question type is determined. Also, named entity are extracted to determine the topic of the question. These topic is searched on Wikipedia to retrieve the content. This forms the document retrieval step. In the last step, the answer is extracted from the content based on the question type. The details of each steps are given in the following subsections

A. *Question Processing*

The Question Processor module consists of two main functions - getQType and getMovieName. The getQType tries to identify the type of the question asked by comparing the tokens with the standard question tokens of ‘Who’, ‘When’, ‘What’, ‘How’. Currently the system only does a simple match with the word and allocates the type of that question depending on the token found. Additionally to further help determine the exact answer, it also tries to find a subtype of the question. This is mainly required for the ‘Who’ and ‘How’ categories. This is done using the Spacy library’s similarity function which allows to compare the query with a fixed corpus or document to determine how similar the query is to the pre decided document. This helps in categorizing whether the question is about a director or an actor or the budget. This is further used as extract answer from the Wikipedia content as detailed in the Section C.

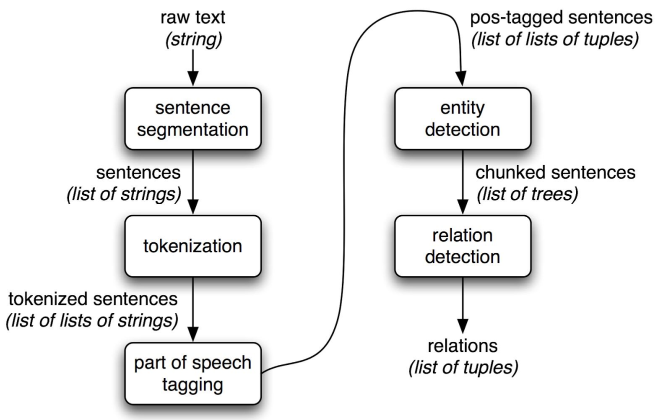
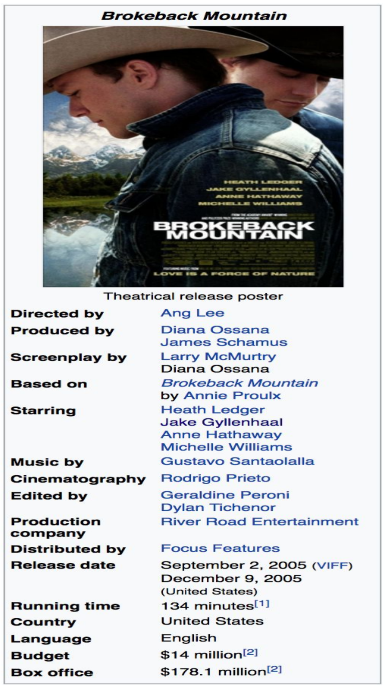


Fig. 2 The entity extraction flow

The movie name is extracted using a combination of Regex, Chunking and custom Chunking grammar rules. If the regex method is unable to directly extract the movie name, then the chunking and custom grammar chunking methods are applied to identify the name. The regex and grammar chunking methods work well for well formed English sentences. Even though it’s not the most perfect method, since most of the factual questions have a fixed format, this works in this methods favour.

B. *Answer Content Fetching*

The system uses wptools Python library to fetch the Wikipedia document corresponding to the topic (movie name) extracted by the getMovieName method. This content is returned in a JSON format. The system mainly relies on the infobox information on Wikipedia. This is a semi-structured content box which is easy to parse and extract information from.



Brokeback Mountain

Theatrical release poster

Directed by	Ang Lee
Produced by	Diana Ossana James Schamus
Screenplay by	Larry McMurtry Diana Ossana
Based on	*Brokeback Mountain* by Annie Proulx
Starring	Heath Ledger Jake Gyllenhaal Anne Hathaway Michelle Williams
Music by	Gustavo Santaolalla
Cinematography	Rodrigo Prieto
Edited by	Geraldine Peroni Dylan Tichenor
Production company	River Road Entertainment
Distributed by	Focus Features
Release date	September 2, 2005 (VIFF) December 9, 2005 (United States)
Running time	134 minutes[1]
Country	United States
Language	English
Budget	$14 million[2]
Box office	$178.1 million[2]

Fig. 3 A typical movie info box on Wikipedia

Once the content is extracted, the actual Wikipedia article link and the infobox content are passed on to the Answer Extraction module.

C. *Answer Processor*

The AnswerProcessor module tries to extract the actual information and answer from the content given by Wikipedia. This makes use of the question type and if available, the subtype. Depending on what is requested in the query, the appropriate regex rules are applied on the appropriate info box field content. This extracts the actual answer in textual format and returns it back to the main process.

The answer, depending on the categories, is extracted from the following fields:

1) *Director*: The movie's director. A simple regex rules is applied to extract the name of the director.

2) *Starring:* The cast of the Movie or the TV series. This a list of names and therefore multiple names are returned..

3) *Budget:* A numeric value to get the budget of the movie.

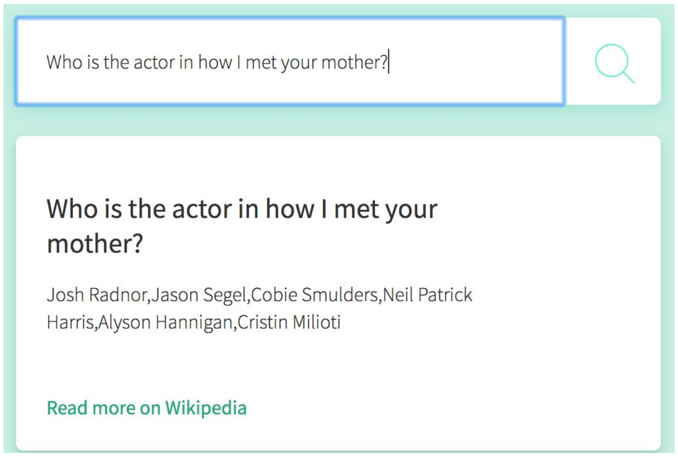
4) *Gross*: The net collection done by movie at the box office.

5) *Released*: The release date of the movie.

IV. Implementation

The whole system is developed using Python. The following third party libraries are the core the system:

1) *nltk*: This has been used heavily throughout the application to do POS tagging, Tokenization, and Chunking.
2) *spacy*: To get the similarity between two sentences as a direct numeric value.
3) *wptools*: To fetch information from Wikipedia using a simple request call.
4) *re*: the inbuilt Python Regex library.
5) flask: The web framework to run the whole web application and provide an easy way to use the application.

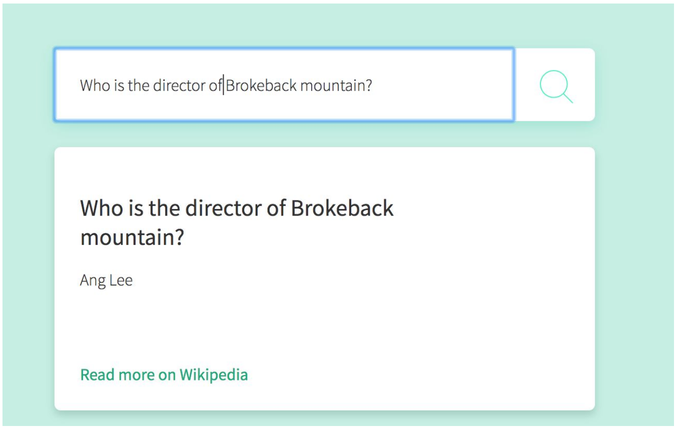

Who is the actor in how I met your mother?

Who is the actor in how I met your mother?

Josh Radnor,Jason Segel,Cobie Smulders,Neil Patrick Harris,Alyson Hannigan,Cristin Milioti

Read more on Wikipedia

Fig. 3 List of Actors in a TV Series

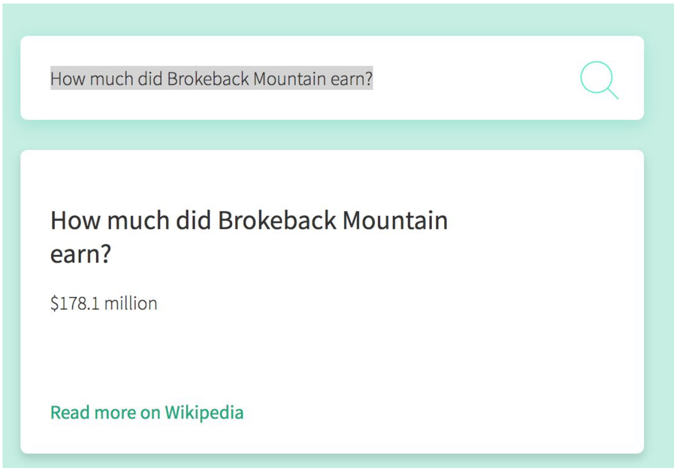

Who is the director of Brokeback mountain?

Who is the director of Brokeback mountain?

Ang Lee

Read more on Wikipedia

Fig. 3 Querying the director of a movie


How much did Brokeback Mountain earn?

How much did Brokeback Mountain earn?

$178.1 million

Read more on Wikipedia

Fig. 3 Querying the box office collection of a movie

Acknowledgment

I am grateful to the great work done by eminent scholars like Daniel Jurafsky & James H. Martin. Also, to the authors of the paper of *Learning Question Classifiers,* Xin Li and Dan Roth.

I would also like to take this opportunity and thank Prof. Dan Moldovan and Linrui Zhang for constructing a great course and giving this opportunity. It has been a great learning experience.

References

[1] Li, X., & Roth, D. (2002). Learning question classifiers. Proceedings of the 19th international conference on Computational linguistics -. doi:10.3115/1072228.1072378

[2] Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright 2017. Draft of August 7, 2017.

[3] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature poly-Si TFT," *IEEE Electron Device Lett.*, vol. 20, pp. 569–571, Nov. 1999.

[4] Mishra, M., Mishra, V. K., & H.r., S. (2013). Question Classification using Semantic, Syntactic and Lexical features. International journal of Web & Semantic Technology, 4(3), 39-47. doi:10.5121/ijwest.2013.4304

[5] Extracting Information from Text, http://www.nltk.org/book/ch07.html

[6] Language Processing and Python, http://www.nltk.org/book/ch01.html

[7] Categorizing and Tagging Words, http://www.nltk.org/book/ch05.html

[8] Learning to Classify Text, http://www.nltk.org/book/ch06.html

[9] Spacy documentation, https://spacy.io/usage/spacy-101

[10] WPTools, https://github.com/siznax/wptools/wiki

[11] *Complete guide to build your own Named Entity Recognizer with Python, http://nlpforhackers.io/named-entity-extraction/*

[12]