Nerdle: Topic-Specific Question Answering Using Wikia Seeds

Umar Maqsud, **Sebastian Arnold**, **Michael Hülfenhaus** and **Alan Akbik**
Database Systems and Information Management Group
Technische Universität Berlin
Einsteinufer 17, 10587 Berlin, Germany

Abstract

The WIKIA project maintains wikis across a diverse range of subjects from areas of popular culture. Each wiki consists of collaboratively authored content and focuses on a particular topic, including franchises such as "Star Trek", "Star Wars" and "The Simpsons". In this paper, we investigate the use of such wikis to create Question-Answering (QA) systems for a given topic. Our key idea is to use a wiki as seed to gather large amounts of relevant text and to use semantic role labeling (SRL) methods to extract N-ary facts from this data. By applying our method to very large amounts of topically focused text, we propose to address the coverage issues that have been noted for QA systems built using such techniques. To illustrate the strengths and weaknesses of the proposed approach, we make a Web demonstrator of our system publicly available; it provides a *QA view* that enables users to pose natural language questions to the system and that visualizes how questions are interpreted and matched to answers. In addition, the demonstrator provides a *graph exploration view* in which users can directly browse the fact base in order to inspect the scope of the extracted information.

1 Introduction

The WIKIA project operates the largest network of collaboratively authored wikis, consisting of over 390.000 wikis on subjects such as games, entertainment and lifestyle, and is available online at *www.wikia.com*. Each wiki is focused on one particular topic and may consist of tens of thousands of pages of text content. Such data, we argue, provides a unique opportunity for extracting structured relational data confined to one domain of interest.

In this paper, we investigate such an approach; our overall goal is to automatically create Question Answering (QA) systems that are "experts" in one field of interest. Our key idea is to use wikis as seeds to a focused crawler to gather as much text as possible for a given topic. The more text we can gather, the greater the chance that we can address coverage issues that related works in QA have noted: Phenomena such as coreferences, synonyms and paraphrasing may negatively affect a QA systems ability to find answers to questions that are phrased differently from occurrences in text (Fader et al., 2013; Ravichandran and Hovy, 2002). By gathering large amounts of topically relevant text and by employing semantic role labeling (SRL) as a means for fact extraction and representation (Shen and Lapata, 2007), our hypothesis is that we can sidestep many of these issues.

The main purpose of this demonstration is therefore to illustrate and discuss in how far a direct application of SRL to domain-specific text can be employed to create a QA system. To this end, we make publicly available a Web demonstrator of a QA system on three topics of popular culture, namely the "Star Trek", "Star Wars" and "The Simpsons" franchises. Our system, named NERDLE, supports eight types of questions, examples of which are given in Table 1. The fact base on these topics is gathered with our proposed approach. The demonstrator offers two views that highlight different aspects of the system: The *QA view* allows users to pose natural language questions and visualizes how questions are

Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations, pages 81–85, Dublin, Ireland, August 23-29 2014.

	STAR TREK	THE SIMPSONS	STAR WARS
WHO	Who attacked the Enterprise?	Who is Homer's father?	Who destroyed the Death Star?
WHERE	Where was Picard born?	Where was Homer born?	Where was the Death Star destroyed?
WHEN	When did James Kirk die?	When was Homer born?	When was Vader killed?
HOW	How did James Kirk die?	How does Homer work?	How did Luke destroy the Death Star?
WHOM	Whom did the Klingons attack?	Whom does Homer know?	Whom did Luke Skywalker attack?
WHAT	What are Klingons?	What does Bart think?	What is the Rebel Alliance?
WHY	Why was Voyager destroyed?	Why was Homer sad?	Why does Luke Skywalker die?
WHICH	Which captain was born on Earth?	Which food critic was born in Springfield?	Which Jedi attacked the Death Star?

Table 1: The eight question types currently supported by our system and example questions for each of the three topics.

interpreted and how they are matched to answers. The *graph exploration view* enables users to directly browse the fact base in order to inspect the scope of the extracted information.

In the following sections, we briefly describe our method for using WIKIA to gather text data for a given topic. We discuss how we employ SRL for fact extraction and representation and give details on our method for aligning eight types of natural language questions to these facts. Finally, we discuss the visualization and results of the created QA system.

2 Method

In this section, we describe our method for gathering relevant text using a wiki seed and extracting facts using SRL. As a running example, we use the "Star Trek"-universe as the topic of interest.

2.1 Constructing Topic-Specific Corpora

As a first step, we select the appropriate wiki for the given topic and retrieve all its text as well as the names of the page titles. For the "Star Trek" example, the WIKIA project offers two wikis, namely *Memory Alpha* and *Memory Beta*[1], both of which we select as relevant. Using these wikis, we download over 95.000 pages or 250 MB of text content for the domain of interest.

We then determine combinations of search keywords using the page titles and use these as queries in search engines like BING[2] or FAROO[3]. Examples of such queries are "Kirk Spock Star Trek" or "Picard Data Star Trek". While FAROO (unlike BING) does not limit the number of allowed queries per month, its index is much smaller. We therefore developed the following strategy to use the FAROO index in our text gathering effort: For each combination of search keywords, we retrieve all matching pages and then follow their outgoing links to find more possibly related Web pages. We check each Web page to contain at least one mention of the domain keyword ("Star Trek") as a simple sanity check to ensure that the crawler has not left the domain of interest. We download all pages we reach with this method.

Using this method, we find over 500 MB of text for the "Star Trek"-domain. As this is an ongoing effort, the corpus size is expected to expand further. The generated corpus is then passed to the fact extraction step of the pipeline.

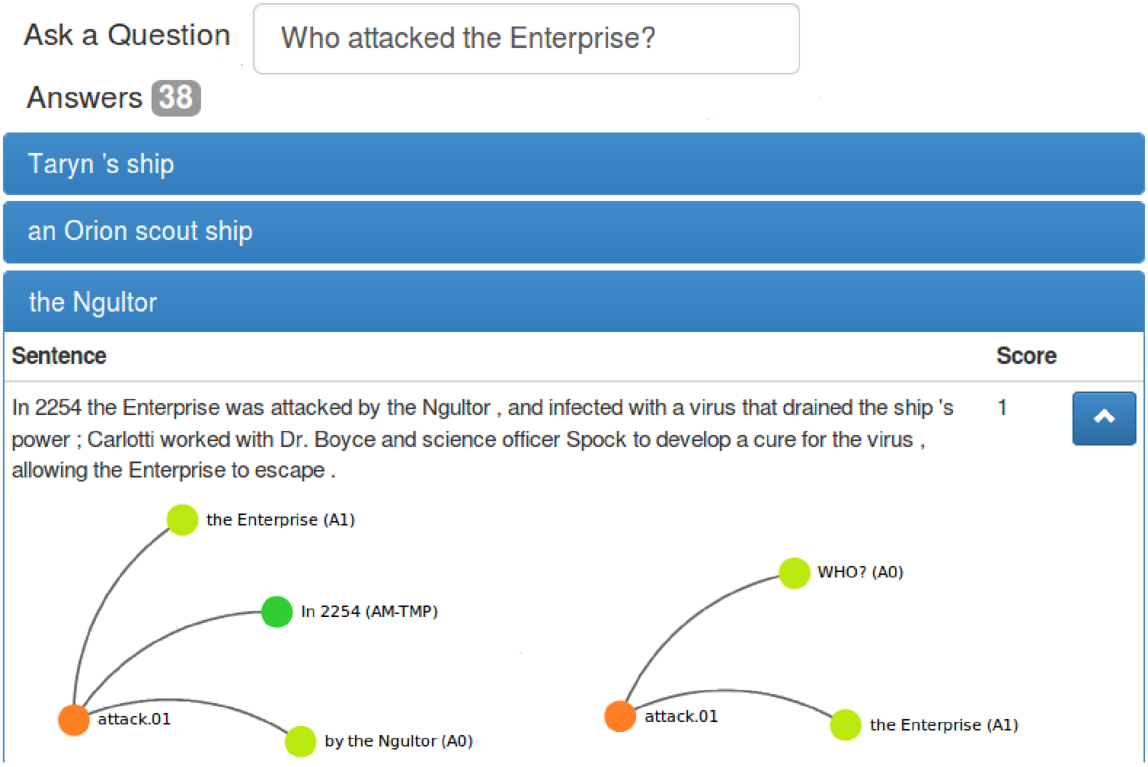
2.2 Fact Extraction

We detect English language sentences in the gathered corpus and apply SRL to detect predicate-argument structures for each verb. We use the ClearNLP toolkit (Choi and Adviser-Palmer, 2012) for this task; it links each predicate to a PROPBANK (Martha and Palmer, 2002) verb sense and its arguments to PROPBANK semantic roles. We choose PROPBANK over FRAMENET (Baker et al., 1998) as it models semantics more broadly and has a more complete coverage of verb frames. For our purpose, we are especially interested in the argument roles: PROPBANK gives us verb-specific argument roles as well as universal roles such as temporals (TMP), locatives (LOC), causal adverbials (CAU) and adverbials

[1] Available at http://en.memory-alpha.org/ and http://memory-beta.wikia.com/ respectively.

[2] http://www.bing.com

[3] http://www.faroo.com/



Ask a Question: Who attacked the Enterprise?

Answers 38

- Taryn 's ship
- an Orion scout ship
- the Ngultor

Sentence	Score
In 2254 the Enterprise was attacked by the Ngultor , and infected with a virus that drained the ship 's power ; Carlotti worked with Dr. Boyce and science officer Spock to develop a cure for the virus , allowing the Enterprise to escape .	1

Figure 1: The QA-view of the Web demonstrator. The user types in a question and receives a list of arguments as answers below. For the question "*Who attacked the Enterprise?*" the user receives a total of 38 answers, three of which are shown here, namely "*Taryn's ship*", "*an Orion scout ship*" and "*the Ngultor*". By clicking on one of the answers, the user inspects the predicate-argument structures of both question (lower right graph) and the answer (lower left graph), as well as the sentence in which the answer is found.

of manner (MNR) and purpose (PRP). In this work, we treat the predicate-argument structures as N-ary facts and store these in a graph database.

To illustrate the fact extraction process, consider the sentence "*In 2254, the Enterprise was attacked by the Ngultor*". Using SRL, we determine a predicate-argument structure in which "attack" is the predicate and there are three arguments: Two arguments with verb-specific roles, namely "*by the Ngultor*" (attacker) and "*the Enterprise*" (that which is attacked), as well as the argument "*in 2254*", which is recognized to be of type *AM-TMP*, meaning that it confers additional temporal information to the ternary fact. This predicate-argument structure is illustrated in Figure 1.

2.3 Question Parsing

The question parsing process is similar to fact extraction; we apply SRL to a question to determine its predicate-argument structure. We then try to find matching predicate-argument structures in the fact base by searching for facts that share the same predicate and as many arguments as possible. The greater the number of matching arguments, the higher the score of the matching fact. In addition, we require matching facts to also contain an *answer argument* labeled with a specific semantic role which is determined through the question type.

We allow seven basic types of questions and one type of composite question. The basic question types we support are factual questions beginning with the question words "who", "where", "when", "whom", "what", "how" and "why". Depending on the question type, we require answer arguments to be labeled with a different semantic role. For "where"-questions, for example, we require answer arguments to be labeled as an *AM-LOC* argument. For "when"-questions, the answer argument must be labeled as an *AM-TMP* argument. For "who"-questions, we require an argument that shares the semantic role of the question word, which typically will be either *A0* or *A1*. These answer arguments are returned answers to the question.

We illustrate this in Figure 1. The question "*Who attacked the Enterprise?*" is parsed into a predicate-argument structure. It is aligned with the predicate-argument structure of the sentence "*In 2254 the Enterprise was attacked by the Ngultor*" because they share the same predicate as well as the argument "*the Enterprise*" with the role *A1*. Of the two remaining arguments, namely "*in 2254*" and "*the Ngultor*" the latter is selected as the answer argument because it carries the same semantic role as was assigned to the token "who" in the question, namely *A0*. In case of a "when"-question, "*in 2254*" would instead be selected as answer argument, as it is labeled with the required *AM-TMP* role.

In addition, we support one type of composite question, namely questions beginning with "which", as in "*Which captain was born on Earth?*". Such questions are decomposed into two separate "who"-questions, namely "*Who is a captain?*" and "*Who was born on Earth?*". We then determine answer arguments that match both questions and return these.

3 Demonstration

We present a Web demonstrator [4] in which users can query the fact base in one of two views:
QA view. In this view, users pose natural language questions and are presented with matching answers if they exist. For each answer, both the source sentences as well as the URLs to the original Web pages are displayed. Answers are ranked by a score which is determined through the number of matching arguments between the predicate-argument structures of the question and the answer. As illustrated in Figure 1, users inspect a visualization of these predicate-argument structures. This view is primarily designed to aid with understanding issues with *precision*, i.e. to understand how answers to questions come to be and how fact extraction and question parsing function.
Graph exploration view. In this view, users can browse the graph database directly for facts. Together with the QA view, this view is designed to examine issues of *recall*, i.e. to help understand the scope of the extracted information and why some questions are not answered.

4 Discussion

With our method, we find a total of 7 million facts for "Star Trek", 6.5 million for "Star Wars" and, due to its smaller wiki size, 3.5 million for "The Simpsons". Next to the availability of large amounts of Web text, our focus on topics of popular culture has the advantage that there are a large number of resources available online that can be used to analyze the QA capabilities of our system. In our analysis, we make use of ABSURDTRIVIA[5], a community powered Web site where users write and rate trivia quizzes on items of popular culture. The trivia quizzes consist of a set of multiple-choice questions. We crawl 50 of these questions on the NERDLE topics that conform to our question types and pose them to NERDLE. We find that NERDLE chooses the correct answer for 16 questions, a wrong answer for 5 questions and no answer at all for 29 questions.

Our demonstrator allows us to inspect wrong and unanswered questions. We find that the system often either lacks the correct facts in the fact base or cannot align questions to answers due to problems of synonymy, entailment and coreferences. An example of this is the question "*Who played Phlox?*" to which no answer is found, while the correct answer is found for "*Who portrayed Phlox?*". This suggests that the coverage of the system might be improved by adding knowledge on synonymous arguments as well as synonymous or entailing verbs. Future work will accordingly examine how synonyms and entailment could be added to improve the coverage of the system. One idea is to leverage wiki page links to identify synonymous entities similar to the work presented in (Spitkovsky and Chang, 2012). In addition, we will expand our crawling efforts to gather larger text corpora and add more question types to the question parser.

Future work will continue to emphasize the visualization of question parsing and answer alignment in order to aid discussion with the research community about the strengths and limitations of SRL for QA.

[4] The demonstrator is available online at http://www.textmining.tu-berlin.de/nerdle/

[5] http://www.absurdtrivia.com

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. Umar Maqsud, Sebastian Arnold, Michael Hülfenhaus and Alan Akbik received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no ICT-2009-4-1 270137 'Scalable Preservation Environments' (SCAPE).

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Jinho D Choi and Martha Adviser-Palmer. 2012. Optimization of natural language processing components for robustness and scalability.

Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL (1)*, pages 1608–1618.

Paul Kingsbury Martha and Martha Palmer. 2002. From treebank to propbank.

Deepak Ravichandran and Eduard Hovy. 2002. 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.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *EMNLP-CoNLL*, pages 12–21. Citeseer.

Valentin I Spitkovsky and Angel X Chang. 2012. A cross-lingual dictionary for english wikipedia concepts. In *LREC*, pages 3168–3175.