SHAPAQA: Shallow Parsing for Question Answering on the World Wide Web

Sabine Buchholz[1]
[1]ILK, Tilburg University
P.O. Box 90153, 5000 LE Tilburg
The Netherlands
s.buchholz@kub.nl

Walter Daelemans[1,2]
[2]CNTS, University of Antwerp
Universiteitsplein 1, 2610 Wilrijk
Belgium
walter.daelemans@uia.ua.ac.be

Proceedings Euroconference Recent Advances in Natural Language Processing (RANLP 2001) 47-51

Abstract

We introduce SHAPAQA, a shallow parsing approach to online, open-domain question answering on the WorldWideWeb. Given a form-based natural language question as input, the system uses a memory-based shallow parser to analyze web pages retrieved using normal keyword search on a search engine. Two versions of the system are evaluated on a test set of 200 questions. In combination with two back-off methods a mean reciprocal rank of .46 is achieved.

1 Introduction

This paper describes SHAPAQA, an online system[1] for open-domain question answering (QA) on the WorldWideWeb (WWW) that uses shallow parsing. Unlike Information Retrieval, question answering does not return documents but answers. To a question like "When was the telephone invented?" it might just return: "The telephone was invented **in 1876**."

The WWW is especially suited for open-domain question answering because it contains many answers to all sorts of questions. In fact, it might even contain too many answers. Different documents may provide contradicting information, by mistake, as part of fiction, or due to different beliefs of the authors. Some seemingly simple questions do not even have one simple answer. e.g. "Who was President of Costa Rica in 1994?": Calderón was until 8th May, after that it was Figueres. Therefore, SHAPAQA does not attempt to return *the* best answer. Rather, it returns a list of all answers found, sorted by frequency, so that users can see what the majority opinion is, and judge for themselves what to think of the minority ones.

SHAPAQA uses shallow parsing to extract exactly those few words that constitute the actual answer (e.g. "1876"). Current shallow parsing techniques do not achieve perfect results. An additional advantage of the frequency approach is that it is not only robust against deviant content of documents, but also against occasional parsing errors, as the answers extracted by those mostly have low frequency. Parsing is not only error-prone but also time-consuming. In designing SHAPAQA, we put special effort into avoiding unnecessary parsing steps.

This paper is organized as follows. Section 2 describes SHAPAQA's architecture. Section 3 reports on the evaluation: preparation of the test set, scoring, results, and comparison to alternative methods. It also introduces a simple way of combining several systems with different "degrees" of NLP, which works best. Section 4 describes related research. Finally, Section 5 summarizes our conclusions.

2 System Architecture

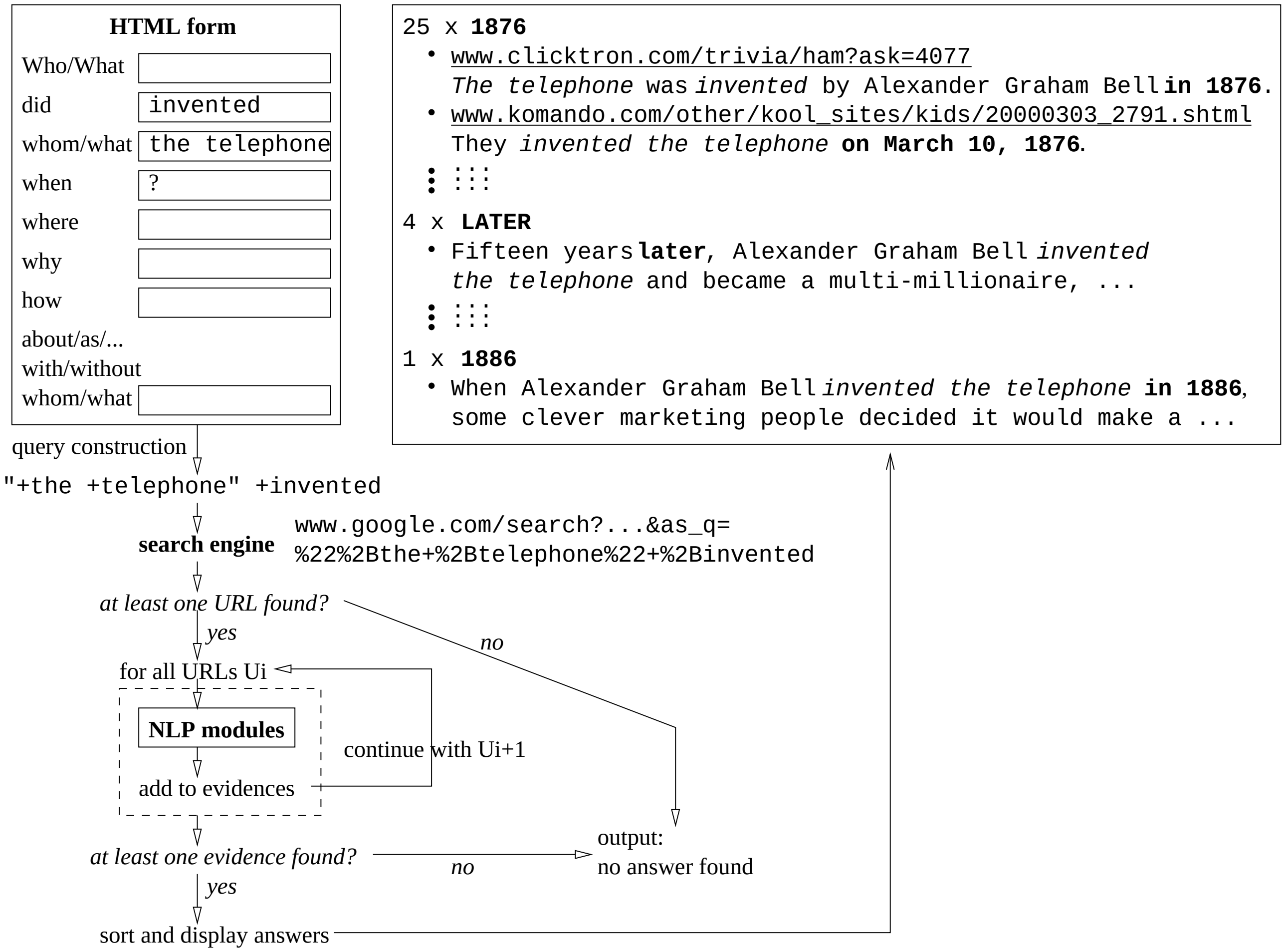
2.1 Example

Suppose we want to know when the telephone was invented. At the moment, questions cannot be entered into SHAPAQA as full natural language sentences but have to be split up into *phrases* by the user. Each phrase is entered into its own HTML form text box. In future versions of SHAPAQA, this task will be performed by a question parser. The top left part of Figure 1 shows the question in its *formatted form*. The parts we know (the *given* phrases) are entered into the appropriate boxes. The parts we are looking for (in this case "when") are indicated through question marks. All the other boxes are left empty. SHAPAQA's results are shown in the top right part of Figure 1. Results consist of *keyword* answers (all capital letters; always one word e.g. "1876"), the number of supporting evidence found (e.g. 25) and the *evidence list*. The evidence list consists of pairs of a URL and a *supporting sentence* found at that URL. In the supporting sentence, the given phrases and the actual answer (the *key chunk*) are highlighted by italic resp. bold font. Keyword answers are sorted by descending frequency as the most frequent answer should have the highest chance of being correct.

2.2 Overall Architecture

The remainder of Figure 1 shows SHAPAQA's overall architecture. SHAPAQA first transforms the question phrases into the search engine query which is submitted to the search engine Google (http://www.google.com/) in "url-encoded" form and the search results are retrieved by SHAPAQA. If some URLs are indeed returned, these are processed one at a time by the NLP modules described below. If the NLP modules find a supporting sentence at the URL, the sentence and the URL are added to the evidence list of the appropriate keyword. Then the next URL is processed. After all URLs have been processed, SHAPAQA checks whether at least some keyword answers

[1] http://ilk.kub.nl/shapaqa/



```
25 x 1876
  • www.clicktron.com/trivia/ham?ask=4077
    The telephone was invented by Alexander Graham Bell in 1876.
  • www.komando.com/other/kool_sites/kids/20000303_2791.shtml
    They invented the telephone on March 10, 1876.
  • ...

4 x LATER
  • Fifteen years later, Alexander Graham Bell invented
    the telephone and became a multi-millionaire, ...
  • ...

1 x 1886
  • When Alexander Graham Bell invented the telephone in 1886,
    some clever marketing people decided it would make a ...
```

Figure 1: SHAPAQA's overall architecture with example input and (partial) output

were found. If so, the answers are sorted and presented to the user.

2.3 NLP modules

The architecture of SHAPAQA's NLP modules is designed to minimize time consuming higher-level NLP as much as possible. Several tests are performed on the data and whenever a test returns "no", this part of the data is not processed any further. The first NLP module is a simple, rule-based tokenizer which processes the text snippet returned by Google together with a URL until it finds a sentence boundary, then lets this sentence be processed by the later tests and modules. Only if one of the tests fails does the tokenizer proceed to find the next sentence.[2] During tokenization, SHAPAQA already stores which words (if any) are part of the given phrases. At the end of a sentence, the first test is whether all given phrases were found. The sentences in (1) did not pass the first test for our example question, although the text as a whole contains all given phrases. (2) shows a sentence that fulfills the test.

(1) The importance of *the telephone* network as a critical factor in the success of fax cannot be overstated. Alexander Bain *invented* the fax machine in ...

(2) *The telephone* was *invented* by Alexander Graham Bell in 1876.

If the test succeeds, a tagger (Daelemans *et al.* 96), a chunker and a module which joins a preposition and one or more (coordinated) NPs into one PNP chunk (Buchholz *et al.* 99) are applied to the sentence. For (2), the output looks as shown in (3) with part-of-speech tags following Penn Treebank conventions (Uni95).

(3) [$_{NP}$ *The*/DT *telephone*/NN $_{NP}$] [$_{VP}$ was/VBD *invented*/VBN $_{VP}$] {$_{PNP}$ [$_{P}$ by/IN $_{P}$] [$_{NP}$ Alexander/NNP Graham/NNP Bell/NNP $_{NP}$] $_{PNP}$} {$_{PNP}$ [$_{P}$ in/IN $_{P}$] [$_{NP}$ 1876/CD $_{NP}$] $_{PNP}$} ./.

SHAPAQA then tests whether the last word of each given phrase is also the last word (i.e. head) of an appropriate chunk. The sentence in (4) would not pass

[2] Due to truncation of text snippets in Google, a "sentence" can also be a partial sentence. All our NLP modules are robust enough to cope with this problem. The method works the same (but slower) on a URL's full content.

this test, as "telephone" is not the last word of the NP chunk.

(4) $[_{NP}$ Touch/NNP Map/NNP Systems/NNP $_{NP}]$ $[_{VP}$ *invented*/VBD $_{VP}]$ $[_{NP}$ *the/DT telephone*/NN dealer/NN locator/NN $_{NP}]$ $\{_{PNP}$ $[_{P}$ over/IN $_{P}]$ $[_{NP}$ seventeen/CD years/NNS $_{NP}]$ $_{PNP}\}$ $[_{ADVP}$ ago/RB $_{ADVP}]$.

Our sentence in (3), however, passes this second test.

2.4 Relation finder

We implemented two versions of SHAPAQA, that differ only in the last NLP module. These are called SHAPAQA GR (grammatical relations) and SHAPAQA CT (chunk types). In SHAPAQA GR, the last NLP module is the relation finder, which determines grammatical relations (like subject, object, temporal modifier) between a verb and other chunks. For each given phrase, it is tested whether this phrase has indeed the relation to the verb indicated by the user. As soon as a given phrase does not have the correct relation, SHAPAQA GR stops processing the sentence. The sentence in (5) did not pass this (third) test, as *the telephone* is not the subject of passive *invented.*

(5) *Invented* at almost the same time as *the telephone* to speed data analysis for the 1880 U.S. Census, the tabulating machine was an electromechanical device that ...

For our example in (3), it would be checked whether *the telephone* is a subject of the passive verb *invented*, which indeed it is. Once all given phrases are found to have the required relation, SHAPAQA GR starts looking for the answer by checking the relations of the chunks surrounding the verb, first the nearest ones, then further away, if necessary up to the first and the last chunk of the sentence. The sentence in (6) did not pass this (fourth) test: no temporal modifier to the verb could be found.

(6) One year after *the telephone* was *invented*, it's usage was taxed.

In (3), the PNP chunk "in 1876" has the right relation and so this chunk is marked as a key chunk, and the sentence added as an evidence under the keyword "1876" (which is the chunk's head).

Relation finding is done by a publicly available machine learning algorithm, the memory-based learner IGTree.[3] Table 1 shows the three *instances* (the rows of the table) derived from our example sentence, one for each pair of a verb chunk and another chunk (the *focus*). Each instance consists of 14 *features* (the columns of the table) and one *class* (the relation). *Feature values* can be numerical, like feature 1, the distance in chunks between the verb and the focus (negative if focus is left of verb). Or values can be symbolic, like feature 2, the verb itself. The focus and the chunk to its left and to its right are each represented by four features: the preposition (in case of PNP chunks), the head word, its POS, and the syntactic chunk type (if any).

The training material for the relation finder was derived from the Wall Street Journal Corpus of the Penn Treebank II (Uni95). To do this, we had to define chunks on the basis of the annotated parse trees, define head words of syntactic constituents, and inherit the labels of a syntactic constituent to its head chunk (i.e. the chunk containing the head word). After being trained on the treebank instances, the learner can assign classes (representing grammatical relations) to new instances in the same format derived from the web pages. More information about the relation finder can be found in (Buchholz *et al.* 99).

2.5 SHAPAQA Chunk Type

Whereas SHAPAQA GR looks for subjects, objects, locative or temporal modifiers etc. *of the verb*, SHAPAQA CT defines the classes NPs, locative or temporal expression etc. independently of any other part of the sentence. As there may be several chunks with the same type in one sentence, the same sentence can be evidence for several keyword answers. The instances are simpler, they just consist of the four features for the focus chunk. Our definition of chunk types overlaps only partially with the concept of Named Entity (NE) types, as used in many TREC systems (cf. Section 4). First, also non-names like "the man" or even non-entities like "later" get chunk types. Second, the common NEs PERSON and ORGANIZATION are not differentiated by chunk types. Third, a place name like "Berlin" would always be of NE type LOCATION, whereas it might be of chunk types LOCATION, OTHER-PP or NP depending on whether it occurs as "in Berlin", "of Berlin" or plain "Berlin".

3 Evaluation

For evaluation, we used the 200 questions from the TREC-8 question answering track (Voorhees & Harman 00), see also Section 4. These are fact-based, short-answer, natural language questions.

3.1 From natural language to form-based questions

The first step of the evaluation was to manually convert the natural language questions into SHAPAQA's question format. While some questions have only one, very obvious "format" (like our old telephone example), others have several. Thus in these cases, results may depend on the particular way of formatting the results. We tried to choose a format that we thought would be used by the average user (given the constraints of the HTML form). The following rules were used:

[3] Software package TiMBL (Daelemans *et al.* 00) available from http://ilk.kub.nl

dist.	verb	left context				focus				right context				class
		prep.	head	pos	chunk	prep.	head	pos	chunk	prep.	head	pos	chunk	
-1	inv.	-	-	-	-	-	tel.	NN	NP	-	inv.	VBN	VP	SBJ
1	inv.	-	inv.	VBN	VP	by	Bell	NNP	NP	in	1876	CD	PNP	LGS
2	inv.	by	Bell	NNP	NP	in	1876	CD	PNP	-	,	,	-	TMP

Table 1: The three instances for the example sentence (some words abbreviated to fit).

- Enter in active form, with the main verb as only verb (cf. our example).
- Skip parts which, when left out, do not change the meaning, like "What is the population of Ulan Bator, capital of Mongolia?"
- Skip verb particles like "up" in "Who came up with the name, El Nino?"
- Format questions with "What is the name of/Name the/How many/Which/What X" as if they were simple "who/what" questions (60 cases)[4]
- Questions with "How far/many times" etc. could not be formatted, so SHAPAQA did not receive any points for them (12 cases).

3.2 Scoring and results

We let SHAPAQA answer the formatted questions, and took the first evidence sentence of each of its top five keyword answers for judging. The human judges then had to read the answers from top to bottom until they found a correct answer to the *original* question[5], say at rank x. The score for this question is then $1/x$ points. The total score of a system is the mean of all the individual scores. This mean reciprocal rank (MRR) metric was also used in TREC. The results are shown in the first two columns of Table 2. We see that SHAPAQA CT performs better than GR.

To put the results into perspective, it is necessary to compare them to other methods of finding answers on the internet. One such method is the search engine Google, which performs keyword search and returns text snippets containing these keywords. We entered all of the words in the formatted version of a test question as keywords into Google, and took the top five text snippets for judging. We also evaluated a variant of SHAPAQA using only the most basic kind of NLP: the sentence tokenizer. If a sentence contained all of the given phrases, it was returned as an answer (this method is henceforth called SENT). Again, top five answers are judged. The results are shown in Table 2: SHAPAQA CT performs better than SENT, and SENT is still better than Google. However, MRR values do not differ dramatically.

The picture changes if we look at the "precision" of the systems: the total points received divided by the number of questions for which the system proposed at least one answer hypothesis. Table 3 shows that the higher the level of NLP used, the less questions a system tries to answer, but for these few, "precision" is higher. This is even true if we compare precision of systems on only those 72 questions that SHAPAQA GR tried to answer. This observation led us to the idea of a combined system: If SHAPAQA GR returned any answers, we took these answers as those of the combined system. If not, and if SHAPAQA CT returned answers, we took those, and so on down to plain Google. As can be seen from the last column of Table 2, the combined system is indeed much better than any of the individual systems. We conclude that this back-off architecture is an easy and succesful way to combine approaches with different degrees of NLP and different "precision" values. Note however that only the two highest approaches (CT and GR) identify the actual answer in the sentence (the key chunk) and therefore allow highlighting and frequency counts.

# quest.	GR	CT	SENT	Google	Combi
200	.28	.34	.32	.30	.46

Table 2: Results over the test questions: system comparison

	GR	CT	SENT	Go.
qu. with ans.	72	101	114	198
points all	55.0	68.4	63.7	60.8
"precision"	.76	.68	.56	.31
points on 72	55.0	52.4	40.4	33.6
"precision"	.76	.73	.56	.47

Table 3: "Precision" of the systems on their answered questions only, and on the 72 answered by GR

Although the difference in precision between the GR and CT versions is not big, there are examples where the former is clearly useful. For the question "Who killed Lee Harvey Oswald?", GR put the correct answer ("Ruby") on top, while CT found "Kennedy" most frequently (and "JFK" third) as it cannot make the difference between subjects and objects, i.e. killers and victims.

4 Related research

Much literature on question answering can be found in the TREC-8 (Voorhees & Harman 00) and TREC-9 proceedings. The three major differences between the

[4] resp. "when" and "where" for "in which year" etc. and "in what city"

[5] Judging largely followed the TREC QA guidelines (Voorhees & Harman 00).

TREC QA task and the evaluation task described here are:

- Systems participating in TREC had to parse the question automatically whereas we formatted it manually. In the future, SHAPAQA will also feature a question parser.
- Answers for the TREC QA track must not excede 50 respectively 250 bytes whereas the answers we evaluated are sentences, which may be longer. However, SHAPAQA GR and CT also identify the key chunk, which is normally much shorter than 50 bytes. When evaluating only the key chunks, MRR is .19 for GR and .26 for CT.
- TREC systems have to find the answer in a given document collection (1904 MB, 528,155 documents) which is guaranteed to contain at least one answer for each question.[6] SHAPAQA works on the WWW, which may or may not contain more answers, but surely contains more noise, so it is unclear whether this makes the task easier or more difficult.

Several TREC systems (Elworthy 01; Scott & Gaizauskas 01; Litkowski 01; Hovy *et al.* 01; Oard *et al.* 00) apply a full parser to the question and potential answer sentences. The more parts of both trees match, the higher the score for a potential answer. SHAPAQA uses only shallow parsing: The relations between words inside the same chunk and beteen two non-verbal chunks (e.g. NP and PP) are not determined. However, *all* of the determined relations have to match. Whereas frequencies are crucial for SHAPAQA, (Singhal *et al.* 00) and (Prager *et al.* 00) are the only ones in TREC to use frequencies of answers as a criterion for answer ranking.

The START system (Katz 97) is an online QA system.[7] It uses a full parser to analyze questions and sentences in text. However, texts are parsed at indexing time and the resulting representations are stored in a knowledge base. There are knowledge bases for many different but certainly not for all domains. As START relies heavily on lexical information, adapting to a new domain probably also means updating the lexicon. SHAPAQA on the other hand parses text at query time and all of its modules can handle unknown words. In principle it can answer questions from any domain for which there are pages on the WWW.

5 Conclusion

In this paper, we described an approach to online, open-domain question answering on the WWW that makes use of a memory-based shallow parser to analyze the relevant parts of documents found with normal keyword search. The main research result is that the use of higher levels of NLP increases the precision of question answering, and leads to higher accuracy (as measured with MRR) when combined with more general systems as back-off.

In the future, full natural language questions will be accepted. This means that we need a question parser that analyzes the question and determines the given phrases and their relations. We also need a way to handle the common “how” and “which/what X” questions.

Acknowledgements

We would like to thank our colleagues who helped judging the many answers. This research was done in the context of the “Induction of Linguistic Knowledge” research programme, which is funded by the Netherlands Organization for Scientific Research (NWO).

References

(Buchholz *et al.* 99) Sabine Buchholz, Jorn Veenstra, and Walter Daelemans. Cascaded grammatical relation assignment. In Pascale Fung and Joe Zhou, editors, *Proceedings of EMNLP/VLC-99*, pages 239–246. ACL, 1999.

(Daelemans *et al.* 96) W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. MBT: A memory-based part of speech tagger generator. In E. Ejerhed and I. Dagan, editors, *Proc. of Fourth Workshop on Very Large Corpora*, pages 14–27. ACL SIGDAT, 1996.

(Daelemans *et al.* 00) Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. TiMBL: Tilburg memory based learner, version 3.0, reference guide. ILK Technical Report 00-01, Tilburg University, 2000. available from http://ilk.kub.nl.

(Elworthy 01) D. Elworthy. Question answering using a large nlp system. In *Proceedings of TREC-9*. NIST, 2001.

(Hovy *et al.* 01) E. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C-Y Lin. Question answering in webclopedia. In *Proceedings of TREC-9*. NIST, 2001.

(Katz 97) Boris Katz. From sentence processing to information access on the world wide web. In *AAAI Spring Symposium on Natural Language Processing for the World Wide Web*, 1997.

(Litkowski 01) K.C. Litkowski. Syntactic clues and lexical resources in question-answering. In *Proceedings of TREC-9*. NIST, 2001.

(Oard *et al.* 00) Douglas W. Oard, Jianqiang Wang, Dekang Lin, and Ian Soboroff. TREC-8 experiments at maryland: CLIR, QA and routing. In *Proceedings of TREC-8*, pages 623–636, 2000.

(Prager *et al.* 00) John Prager, Dragomir Radev, Eric Brown, Anni Cohen, and Valerie Samn. The use of predictive annotation for question answering in TREC8. In *Proceedings of TREC-8*, pages 399–410, 2000.

(Scott & Gaizauskas 01) S. Scott and R. Gaizauskas. University of sheffield trec-9 q&a system. In *Proceedings of TREC-9*. NIST, 2001.

(Singhal *et al.* 00) Amit Singhal, Steve Abney, Michiel Bacchiani, Michael Collins, Donald Hindle, and Fernando Pereira. AT&T at TREC-8. In *Proceedings of TREC-8*, pages 317–330, 2000.

(Uni95) University of Pennsylvania, Philadelphia. *Bracketing Guidelines for Treebank II Style, Penn Treebank Project*, 1995.

(Voorhees & Harman 00) E.M. Voorhees and D.K. Harman, editors. *Proceedings of TREC-8*, number 500-246 in NIST Special Publication. National Institute of Standards and Technology (NIST), 2000. available at http://trec.nist.gov/pubs.html.

[6] This condition is abandoned in TREC-10.

[7] http://www.ai.mit.edu/projects/infolab/