Natural Language Engineering **7** (4): 301–323. © 2001 Cambridge University Press
DOI: 10.1017/S1351324901002777 Printed in the United Kingdom

Complex answers: a case study using a WWW question answering system

S. BUCHHOLZ[1]
[1]*ILK Computational Linguistics, Tilburg University, The Netherlands*
e-mail: s.buchholz@kub.nl

W. DAELEMANS[1,2]
[2]*CNTS Language Technology Group, University of Antwerp, Belgium*
e-mail: walter.daelemans@uia.ua.ac.be

Abstract

We investigate the problem of *complex answers* in question answering. Complex answers consist of several simple answers. We describe the online question answering system SHAPAQA, and using data from this system we show that the problem of complex answers is quite common. We define nine types of complex questions, and suggest two approaches, based on answer frequencies, that allow question answering systems to tackle the problem.

1 Introduction

The underlying assumption in much research on Question Answering (QA) is that in most cases there is one (correct) answer to a question, and that this answer has to be found. We think that whether or not a question has exactly *one* answer depends on how one defines an answer. In our view, many answers are *complex* answers, which contain two or more *simple* answers. An example is the question "Who was President of Costa Rica in 1994?".[1] The most complete answer would be something like: "Calderón was president until 8th May, after that Figueres became president." which consists of two parts. The problem of complex answers is relevant when building QA systems, but also when evaluating them. We will treat two aspects:

- What types of complex answers are there and how frequent are they?
- How could systems deal with the problem?

To get real data on this topic, we used a system for QA on the World Wide Web (WWW). As the WWW contains documents from many different times, written in different countries, from different points of view, with different background knowledge assumed, etc. it is particularly suited for finding complex answers. The

[1] Taken from the TREC-8 test question set, cf. Section 2.3 and 3.2.

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

QA system SHAPAQA is introduced in the first half of this article (section 2). We first sketch its overall architecture and design principles. The core of the system, the Natural Language Processing (NLP) modules, are all based on the same machine learning algorithm, Memory-Based Learning (MBL), so this algorithm is explained next (section 2.1). Then the NLP modules themselves are described in section 2.2 and finally the performance of SHAPAQA on a set of test questions is evaluated in section 2.3. The second half of this article (section 3) is dedicated to the analysis of SHAPAQA's results from the perspective of the problem of complex answers. We first present a typology of complex answers in section 3.1. In section 3.2 we give an overview of how complex answers are handled in the QA track of the Text REtrieval Conferences (TREC), for which much work on QA systems and evaluation has been done. Next we suggest two ways to approach complex answers (section 3.3) and finally discuss remaining challenges in section 4.

2 SHAPAQA

SHAPAQA (SHAllow PArsing for QA) is an online QA system for the WWW.[2] Let us first illustrate its functionality through an example. Suppose we want to know when the telephone was invented. At the moment, the question cannot be entered as a full natural language sentence but has to be split up into what we will call *phrases*. Each phrase is entered into its own HTML form text box. The top left part of figure 1 shows the question in its *formatted form*. The parts we know (the *given* phrases; in this case 'invented' and 'the telephone') 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. The information entered in the HTML form could maybe best be paraphrased as "Somebody (I don't care who) invented the telephone somewhere (I don't care where) when?".

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 list of this evidence. 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 answer (the *key chunk*) are highlighted by respectively italics and bold font.[3] Keyword answers are sorted by descending frequency. The lower the frequency, the higher the chance that the answer is due to a parser error or to wrong information in the documents. There is no sorting among supporting sentences; but cf. section 3.3.2.

The remainder of figure 1 shows SHAPAQA's overall architecture. After the user has entered the question into the HTML form and pressed the submit button, SHAPAQA transforms the question phrases into the search engine query. The query is submitted to a search engine[4] in 'url-encoded' form and the search results are retrieved by SHAPAQA. If the search engine does not return any URL for the query, SHAPAQA

[2] `http://ilk.kub.nl/shapaqa/`
[3] In the actual WWW implementation, highlighting is done by colors.
[4] currently Google (`http://www.google.com/`)

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

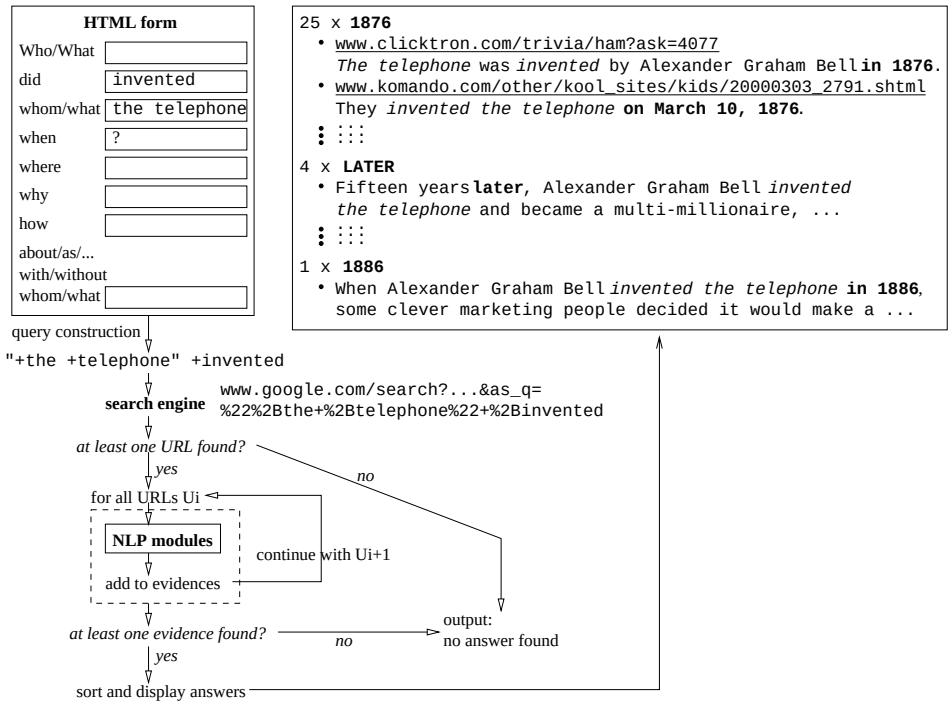


Fig. 1. SHAPAQA's overall architecture with example input and (partial) output.

terminates with an appropriate message. If some URLs are indeed returned, these are processed one at a time by the NLP modules described in section 2.2. If the NLP modules find a supporting sentence in the URL's content, the sentence and the URL are added to the evidence list of the appropriate keyword. The rest of the document is skipped for efficiency reasons, as it is unlikely that it contains another different answer. Then the next URL is processed. After all URLs have been processed, SHAPAQA checks whether at least some keyword answers (with an associated list of supporting evidence) were found. If so, the answers are sorted by (evidence) frequency and presented to the user. If not, SHAPAQA terminates with an appropriate message.

Nearly all of SHAPAQA's NLP modules use a machine learning (ML) algorithm known as Memory-Based Learning as the underlying technology. We will therefore first introduce this algorithm together with general concepts of ML in section 2.1 before describing the actual NLP modules in section 2.2.

2.1 Memory-Based Learning

Memory-Based Learning (MBL) belongs to the group of supervised ML algorithms. Supervised algorithms learn from an *annotated* training set how to process an *unannotated* test set. For MBL to be applied to a learning problem, it has to be formulated as a *classification* problem: the learner (*classifier*) has to assign exactly one out of several *classes* to each input. Take the problem of part-of-speech (POS)

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

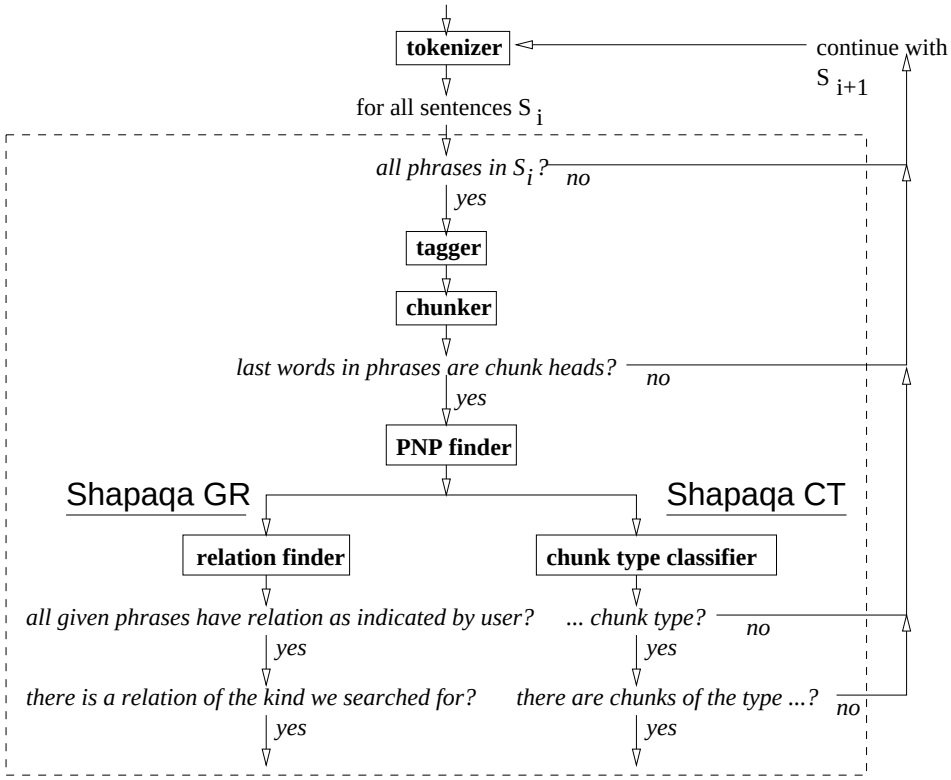


Fig. 2. SHAPAQA's NLP modules.

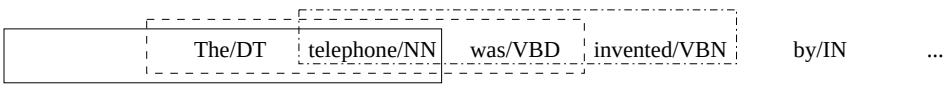


Fig. 3. A three word window moved over the first words of the sentence "The telephone was invented by ..." in order to create instances.

tagging as an example. The inputs correspond to the words and their contexts, and the classes to the proper POS of the word in that context. MBL belongs to the class of propositional learners, so the inputs (*instances*) have to be in a propositional *feature-value* format. A common method for converting a sequence such as a sentence into feature-value vectors is called *windowing*. A fixed size window is moved over the sequence, and (part of) the information that falls into the window is taken as input feature values. Figure 3 shows a three word window that is moved over the start of a sentence and Table 1 shows the corresponding instances.

The four features are the *focus* word (i.e. the word whose class the learner should predict), the previous word (left context) and its POS, and the following word (right context).[5] As tagging proceeds from left to right, the POS of the right context is not

[5] If no word precedes, the dummy value '–' is used instead.

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

Table 1. *The instances for POS tagging for the first three words of the sentence "The telephone was invented by ..."*

Left context		Focus	Right context	
Word	POS	Word	Word	Class
–	–	The	telephone	DT
The	DT	telephone	was	NN
telephone	NN	was	invented	VBN

yet known and can therefore not be used as a feature. The class is the POS of the focus word. The class is given for training instances and sought for test instances.

MBL works on the assumption that for language processing tasks, *lazy learning* has an advantage compared to *eager learning* methods (Daelemans, Van den Bosch and Zavrel 1999) because of the complex interaction of regularities, subregularities and exceptions in language processing tasks. Lazy learning methods, such as MBL, simply store all training set instances. New inputs are classified by similarity-based reasoning on the basis of these stored cases (including exceptions). The class of a new input will be based on the class(es) of those instances in memory that are most similar to it. In contrast, eager learning methods extract general rules or other knowledge structures from the training instances, ignoring exceptional and low-frequency instances in the process.

The similarity metric used in MBL is of the utmost importance. In our approach to MBL, information theory is used to assign relevance to the different features of an instance during similarity computation. In its most naive implementation, MBL is an expensive algorithm (each new input instance to be classified has to be compared to all stored instances). We developed an alternative approach which, while adhering as much as possible to the philosophy of keeping all training data, achieves higher efficiency by restructuring memory in such a way that it contains the same information as before, but in a compressed decision tree structure (the IGTree algorithm, see Daelemans, Van den Bosch and Weijters (1997) for details).[6]

2.2 NLP modules

As SHAPAQA is designed to work online, one of the main requirements is that it be fast. The parts that were described in the beginning of this section (query construction, search engine, answer sorting and display) do not need much time. The bottleneck is the NLP modules, especially the higher-level ones like parsing. The problem is enlarged by the fact that SHAPAQA's answer sorting is based on frequencies and that frequency counts are known to become unreliable when using too small a sample. Therefore, SHAPAQA uses the top 1000 URLs returned by the search engine. It should be clear that we cannot fully parse 1000 documents online with current technology.

[6] The software package TiMBL (Daelemans, Zavrel, van der Sloot and van den Bosch 2000) incorporating IGTtree and other MBL algorithms, is available from `http://ilk.kub.nl`.

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

Fortunately, we need neither to parse complete documents nor to have a full parse. Instead we shallow parse selected sentences. As even retrieving 1000 documents might take quite some time, we use the text snippets returned by Google for a URL as a short-cut (although we would prefer to have a search engine that returns whole sentences). As the text snippets have a fixed size and frequently even contain pieces from several places in the document (separated by ellipsis dots), the parser has to be able to cope with partial sentences. These considerations led to the architecture of SHAPAQA's NLP modules as shown in figure 2. The boxes are the modules. The italic questions in figure 2 are tests on the data. Whenever a test returns 'no', this part of the data is not processed any further. We call this concept *parsing on demand*.

The first NLP module is a simple, rule-based tokenizer. In the spirit of *parsing on demand* the tokenizer just processes a text snippet until it finds a sentence boundary or ellipsis dots, then lets this (partial) 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. 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, Zavrel, Berck and Gillis 1996) and a chunker (Buchholz, Veenstra and Daelemans 1999) using the MBL algorithm introduced in section 2.1 are applied to the sentence. For (2), the output looks as shown in (3) with part-of-speech tags following Penn Treebank conventions (Bies, Ferguson, Katz and MacIntyre 1995).

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

After chunking, SHAPAQA tests whether the last word of each given phrase is also the last word (i.e. head) of an appropriate chunk, e.g. a VP for the given verb. The sentence in (4) would not pass this test, as 'telephone' is not the last word of the NP chunk.

(4) [$_{NP}$ Touch-Map Systems] [$_{VP}$ *invented*] [$_{NP}$ *the telephone* dealer locator] [$_{P}$ over] [$_{NP}$ seventeen years] [$_{ADVP}$ ago] .

Our sentence in (3), however, passes this second test. Chunking serves to make the sentence representation more compact by deleting all non-heads of chunks. Our sentence would then look like (5) .

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

Table 2. *Correspondences between* SHAPAQA*'s input form, classes, and labels in the Penn Treebank*

HTML form	SHAPAQA GR class	SHAPAQA GR treebank	SHAPAQA CT class	SHAPAQA CT treebank
Who/What	SBJ	-SBJ (subject) of active verb	NP	-SBJ, -LGS; NP without -TMP/LOC /PRP/MNR
	LGS	-LGS (logical subject) of passive verb		
whom/what	OBJ	NP (object), NP-PRD (predicative), NP-CLR (closely related) of active verb	NP	same as above
	SBJ	-SBJ of passive verb		
when	TMP	-TMP (temporal) of verb	TMP	-TMP
where	LOC	-LOC (locative) of verb	LOC	-LOC
why	PRP	-PRP (purpose and reason) of verb	PRP	-PRP
how	MNR	-MNR (manner) of verb	MNR	-MNR
about/as/... /with/without whom/what	OTH	PP of verb, without -TMP /LOC/PRP/MNR and with corresponding preposition	OTH	PP without -TMP /LOC/PRP/MNR + corresp. preposition

(5) [$_{NP}$ *telephone*/NN] [$_{VP}$ *invented*/VBN] [$_{P}$ by/IN] [$_{NP}$ Bell/NNP] [$_{P}$ in/IN] [$_{NP}$ 1876/CD] ./.

It is then processed by the PNP finder (Buchholz, Veenstra and Daelemans 1999). This module joins a preposition and one or more (coordinated) NPs into one PP.[7] PNPs allow further compression of the sentence, as shown in (6) .

(6) [$_{NP}$ *telephone*/NN] [$_{VP}$ *invented*/VBN] [$_{PNP}$ by Bell/NNP] [$_{PNP}$ in 1876/CD] ./.

2.2.1 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), respectively. SHAPAQA CT will be described later. In SHAPAQA GR, the last NLP module is the relation finder, which determines grammatical relations between a verb and other chunks using MBL. Possible relations (classes) and their correspondence to the HTML form boxes are shown in Table 2. Relation finding is done on a *parsing on demand* basis: Each *given phrase* (entered by the user) is tested to determine whether it has 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 (7) did not pass this (third) test, as *the telephone* is not the subject of passive *invented*.

[7] Syntactic categories other than NPs (e.g. a VP as in "Thank you for coming.") are not joined at the moment, hence the name PNP finder instead of PP finder.

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

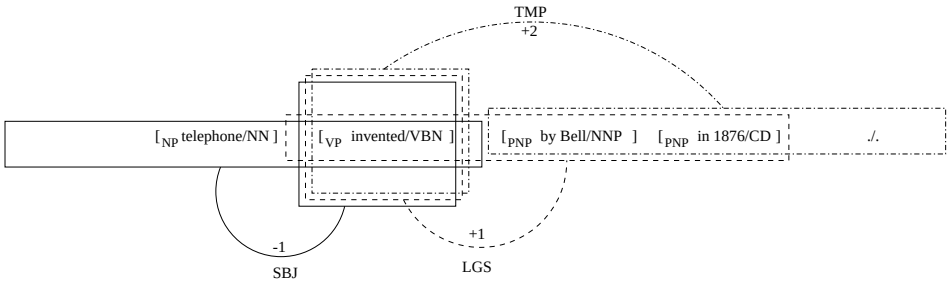


Fig. 4. Two windows moved over the first chunks of the (compressed) sentence "The telephone was invented by Alexander Graham Bell in 1876." to create instances. The numerical values on the arcs are the distance between the centers of the windows, the labels are the grammatical relation.

(7) *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 (6), the system checks 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 (8) did not pass this (fourth) test: no temporal relation to the verb could be found.

(8) One year after *the telephone* was *invented*, its usage was taxed.

In (6), the PNP chunk 'in 1876' has the right relation and so this chunk is marked as a key chunk, and the sentence added as evidence under the keyword '1876' (which is the chunk's head).

The instances for the relation finder are also obtained by windowing (cf. the previous section), only now an instance does not correspond to one word but to a pair (a verb and the head of another chunk) between which the grammatical relation exists. Therefore we have two windows, one of size one moving over all the VP chunks in the sentence and one of size three moving over all the other chunks. This is depicted in figure 4. The resulting instances are shown in Table 3.

The training material for the relation finder was derived from the *Wall Street Journal* Corpus of the Penn Treebank II (Bies, Ferguson, Katz and MacIntyre 1995). 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).[8]

Relation finding as we defined it relies only on local context. When trained on (treebank) full sentences and then applied to partial sentences (from Google text snippets) only the instances having the (false) sentence start/end in their defining

[8] See `http://ilk.kub.nl/~sabine/chunklink/` for the conversion program.

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

Table 3. *The three relation finder instances for the example sentence "The telephone was invented by Alexander Graham Bell in 1876." The features are the distance between the VP chunk and the focus chunk, the verb itself, and the focus chunk with its left and right context chunks (or punctuation tokens), each represented by four features: the preposition (if any), the head word, the POS and the type of chunk (if any)*

Dist.	Verb	Left context				Focus				Right context				Class
		pr.	word	POS	ch.	pr.	word	POS	ch.	pr.	word	POS	ch.	
-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

window might be classified differently whereas in a probabilistic full parser, the probability of the whole parse might be affected. In addition, full parsers like Collins (1997) do not provide us with the necessary information about, for example, different kinds of adjuncts.

2.2.2 SHAPAQA *Chunk type*

As can be seen in Table 2, the main difference between SHAPAQA GR and SHAPAQA CT is that in the latter, no reference to the verb is made in the definition of the classes (which correspond to the types of chunks that are recognized). One of the consequences of this is that no distinction is made between subjects and objects. Both have the class NP. More important however, the chunk need not be a syntactic dependent of the verb: All NPs or temporal expressions in a sentence get the NP or TMP class respectively. Therefore the same sentence can be evidence for several keyword answers. As the definition of chunk types is simpler than that of relations, the instances are simpler too. They just consist of the four features for the focus chunk. The classes of the three simpler instances, for the example in (6), would be NP, NP and TMP.

2.3 Evaluation

Although we are mainly interested in complex answers in this article, we want to give the reader an idea of the performance of SHAPAQA. To this end, we tested it on the 200 fact-based, short-answer, natural language questions from the TREC-8 QA track (Voorhees and Tice 2000), see also section 3.2.

2.3.1 From natural language to form-based questions

The first step of the evaluation was to convert the natural language questions into SHAPAQA's question format. While some questions have only one, very obvious 'format' (like our telephone example), others have several. Thus in these cases, results

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

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:

- Enter in active form, with the main verb as only verb.
- Skip redundant parts 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/How much/Which/What X" as if they were simple "who/what" questions (60 cases).[9]
- Questions with "How far/many times" etc. could not be formatted. Consequently, SHAPAQA did not receive any points for them (12 cases).

2.3.2 Scoring and performance

We let SHAPAQA answer the formatted questions, and took the first evidence sentence of each of its top five keyword answers for judging. From our telephone example in figure 1 we would then take the first of the 25 sentences under the keyword '1876', the first of the four sentences under 'LATER', the one sentence under '1886' and so on (rest omitted in figure). The human assessors had to read the answers from top to bottom until they found a correct answer to the *original* question, 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 (cf. section 3.2).

Judging largely followed the TREC guidelines (Voorhees and Tice 2000), but we used two extra evaluation rules. First, if several pieces of information that together answer the question are spread over several answer ranks, the lowest rank counts. The idea is that users have to read at least that many system answers to know the complete answer. The rule was applied for the question "How many Grand Slam titles did Bjorn Borg win?", with the first answer stating that Borg won six French Opens and the second that he won Wimbledon five times, which together allows us to compute the correct answer (11), and therefore gained the system 1/2 point. In principle, it could also be applied to questions that explicitly ask for two entities ("What two researchers ...?"). However this was not necessary in our data as both entities appeared in the same answer.

Our second special rule says that if only half of the answer (i.e. one of the entities) is among the five system answers, the system receives half of the score for that rank (e.g. 1/6 if half of the answer is on the third rank). The idea is that half the answer is at least better than nothing and should be rewarded accordingly. However, using both rules together yields counterintuitive results: if, for a question that asks for two entities, only one is found, at e.g. the first rank, the system receives 1/2 point (second rule). If it also found the other entity, at e.g. the fifth rank, it would only get

[9] respectively "when" and "where" for "in which year" etc. and "in what city"

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

Table 4. SHAPAQA*'s results over the 200 test questions, single and combined system, and scoring only over answered questions ('precision')*

	Shapaqa GR	Shapaqa CT	Combination
MRR	0.28	0.34	0.36
questions with answers	72	101	
points	55.0	68.4	
'precision'	0.76	0.68	

1/5 point (first rule) i.e. fewer for finding more![10] This problem in fact illustrates the difficulties of evaluating complex answers. We do not think that it influenced our evaluation as the two entities always occurred together.

The results are shown in the first row of Table 4. We see that SHAPAQA CT performs better than GR. One might conclude that relation finding (or at least our implementation) is not useful for QA. There is however another way to look at the results. The tests imposed on sentences by SHAPAQA GR are very restrictive, which means that not many answers are found, but those found are mostly correct. This is similar to the precision/recall trade-off known for many problems. So although SHAPAQA GR received fewer points, and answered fewer questions, its 'precision' (points per question) is higher than that of SHAPAQA CT (see rest of Table 4). This 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. The combined system was not implemented but simulated from the individual system results. As can be seen from the third column of Table 4, the combined system performs a little better than SHAPAQA CT itself. We take this as an indication that grammatical relations in principle are useful for QA. Among the questions for which SHAPAQA GR received more points than CT, the most typical example is "Who killed Lee Harvey Oswald?". In GR, the correct answer ("Ruby") easily came on top, while CT found "Kennedy" most frequently, and "JFK" third, because it has no way to distinguish between a subject and an object of the "killing".

The combined system found a correct answer for about 76 questions[11] (and in 68 cases the correct answer was indeed the first). For 74 questions, URLs were retrieved but no correct answer was found in them (in 49 cases no answer was found, in the others only incorrect ones). We know from our evaluation that for at least 30 of these 74 questions, the answer was in the URLs but SHAPAQA failed to find it (for various reasons). For 50 questions, no URLs were retrieved, either because the question could not be formatted (12 cases) or the given phrases were so specific that no document contained all of them. This preliminary analysis indicates that all components of the system show room for improvement.

[10] Thanks to one of the reviewers for pointing this out.
[11] 'about' because of different opinions among assessors.

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

At present, the accuracy of the shallow parser is around 71% combined precision and recall for all relations (75% precision and 68% recall), as reported in Buchholz, Veenstra and Daelemans (1999). Relations like subject and object reach around 80% combined precision and recall. Although we have not quantified the influence of the accuracy of the relation finder on the overall results, it seems clear that there is room for improvement here as well by improvements in the NLP modules. For example, with perfect chunking information, overall accuracy rises to 81% precision and 74% recall for all relations on average.

We are planning to replace the form based input by a true natural language input facility in the future. We would then need a parser to determine phrases and grammatical relations in the question. Unfortunately, the *Wall Street Journal* on which we trained our shallow parser described above does not contain many direct questions, and therefore the parsers performance on questions is not good enough. We plan to retrain the parser on the parsed Brown Corpus that is included in the third release of the Penn Treebank (Marcus, Santorini and Marcinkiewicz 1993).

3 Complex answers

In the introduction to this article, we cited the following question with a complex answer: "Who was President of Costa Rica in 1994?" "Calderón until 8th May, then Figueres". Complex answers pose a problem for QA systems because the answer need not appear as a coherent string in one document, and for QA system evaluation because it is unclear how to formalize the feeling that the complex answer is 'better' than its parts alone.

3.1 Kinds of complex answers

While developing and evaluating SHAPAQA, we noticed many complex answers. The following list shows the various types we identified:

- Collective answer: "What two US biochemists won the Nobel Prize in medicine in 1992?". "Edmund Fischer and Edwin Krebs won it together". Note that a more realistic question would be "Who won ...?" in which case we cannot know in advance whether the answer is complex. The TREC-8 test set features one other question of this type: "What two researchers discovered the double-helix structure of DNA in 1953?"
- Many answers: "Name a film that has won the Golden Bear in the Berlin Film Festival?" In this formulation of the question, one film should be enough as an answer. However, naming several films feels like a 'better' answer. If the question were "Which films won ...?" we do not know whether the person asking expects to get all films or only some reasonable number of it. To make things even more complicated, one of the many answers might be a collective answer if the prize can be shared.
- Salience/importance: "What does the Peugeot company manufacture?" Peugeot is a big company and produces many things, therefore many answers

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

are possible. However, "the pepper grinder which the company patented last century", although correct, seems to be a less satisfying answer than e.g. "cars".

- Granularity: "Where did Dylan Thomas die?" "At St. Vincents Hospital/New York/United States". This problem appears not only with 'where' questions but also with 'when' ("Tuesday, 19th June 1972 at 2 o'clock" or "in the seventies"), 'how many', 'how long', etc. It is also important for the "Peugeot" question: Is "cars" a better answer than "the Peugeot 405, the Peugeot 309, etc."?

- Different measures: "What was the monetary value of the Nobel Peace Prize in 1989?" "7.6 million Swedish crowns, at that time $960,000" This problem is similar to the previous in that we do not know what granularity/measure the person asking wants.

- Different designations: "What is the brightest star visible from Earth?" "Sirius (A), also known as the Dogstar/Dog Star". We can get 'spurious' different designations from different transliterations of foreign names. Another example is the Costa Rican president "Figueres" who in some text was found as "Figueres Olsen".

- Ambiguity in the question: "What is the capital of Congo?". "Brazzaville is the capital of the Republic of the Congo, and Kinshasa the one of the Democratic Republic of the Congo, the former Zaire". This is the opposite phenomenon of different designations: one name designates different entitites. It is likely to occur with city names ("Where is Brunswick?") or last names of famous people ("Who killed Kennedy?"). In the question "What is the largest city in Germany?", the term "the largest" is ambiguous as the following two answers show: "With an estimated 3.5 million inhabitants, Berlin is the largest city in Germany." "Measuring approximately 23 miles from north to south and 28 miles from east to west, Berlin is by far the largest city in Germany."

- Time dependency: "Who was President of Costa Rica in 1994?" "Calderón until 8th May, then Figueres". A slightly different problem occurs with questions in the present tense like "Who is the prime minister of Japan?" Clearly, this means "Who is prime minister now?", but suppose the prime minister changed, e.g. two days ago. In that case the most informative answer probably still comprises two entities.

- Different beliefs: "Who killed Kennedy?" "Officially the killer was Lee Harvey Oswald, but alternative theories abound, e.g. ..." Different beliefs can be found in great abundance on the internet. It is not easy to draw a line between a different belief and wrong information. The "1886" answer from our telephone example may be classified as wrong, but some of the other dates found pertain to different beliefs about who 'really' invented the telephone (Bell, Reis, Gray?) or when one can speak of the event of invention (when having the idea, the working prototype, the patent?). In any case we cannot expect a QA system to make this distinction.

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

3.2 Complex answers in TREC

Much literature on QA can be found in the TREC-8 proceedings (Voorhees and Harman 2000).[12] The TREC-8 QA track (Voorhees 2000) consisted of automatically answering 200 fact-based, short-answer, natural language questions on the basis of a given document collection of mainly US newspaper articles (1904 MB, 528,155 documents). The collection was guaranteed to contain at least one answer for each question (This condition will be dropped in the upcoming TREC-10 QA track). TREC systems had to return up to five ranked strings of maximally 50 (or 250) bytes each per question. Twenty organizations took part in the QA track. MRR scores (cf. Section 2.3.2) varied from 0.66 to 0.017 with an average of 0.25.[13]

3.2.1 Complex answers in data collection

The data collection used for the TREC-8 QA tracks contains about 500,000 documents. Although this is huge, it is still some orders of magnitude smaller than the WWW. Most of the TREC documents are newspaper articles, many from the US. Consequently the TREC data is also more homogeneous than the WWW. In addition, it only contains documents from 1989 to 1995 whereas the WWW probably features text from about any time (older ones scanned or re-typed). These aspects of the TREC data probably influence the presence of (parts of) complex answers. Fewer documents decrease the probability of finding variations of granularity, many answers for the 'many answer' questions or different answers for ambiguous questions. Homogeneity lessens the use of different measures or of differently spelled designations and the chance for finding different beliefs. Finally, a short time span restricts answers to time dependent questions. We cannot quantitatively compare the presence/absence of (parts of) complex answers on the WWW and the TREC collection, but we will show some questions for which SHAPAQA found more parts (on the WWW) than the TREC-8 systems (in the TREC data).[14] This does not necessarily mean that the parts are not in the TREC data, but may be an indication of it, as about 20 systems did not find them.

- When did the Jurassic Period end? (different beliefs)
  TREC: 130 million years ago vs. WWW: 136/144/approximately 141 million years ago.
- Where was Ulysses S. Grant born? (granularity)
  TREC: Point Pleasant, Ohio vs. WWW: Point Pleasant, Clermont County,

[12] The last TREC conference (TREC-9) also featured a QA track, but the complete proceedings were not available at the time of writing. A larger document collection was used, there were more questions and the questions were less artificial. In addition, some questions were variants of others.

[13] Note that the TREC task is harder than SHAPAQA's as SHAPAQA assumes the questions to be already *formatted*. Whether finding answers in a fixed document collection is more or less difficult than finding them on the WWW is an open question.

[14] TREC-8 system results are taken from the judgement file `qrels.trec8.qa` on `http://trec.nist.gov/data/qrels_eng/index.html`.

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

Ohio/on Earth/in a white Allegheny pine house on the banks of the Ohio River twenty-five miles southeast of Cincinnati.

- Who is the president of Stanford University? (time dependency)
  TREC: Donald Kennedy vs. WWW: John Hennessy, Gerhard Casper (Kennedy is found only as "former president").
- Who played the part of the Godfather in the movie, 'The Godfather'? (ambiguity)
  TREC: – vs. WWW: Marlon Brando, Robert DeNiro (in Part II).
- What is the brightest star visible from Earth? (different designations)
  TREC: Sirius vs. WWW: Sirius, Sirius A, the Dog Star, the Dogstar (plus two documents mentioning that actually the sun is the brightest star).

Despite these examples, complex answers can be found in the TREC data collection. This can be seen in the file containing regular expressions for automatic scoring. The file was created as a service to the community during manual judgement of the track submissions. It can be used together with a scoring program to (roughly) score the output of a system. If a regular expression for a question (number) matches the output for that question, that output is scored correct. The file contains multiple regular expressions for 69 of the 200 TREC-8 test questions, which means that the assessors found several alternative correct answers.

3.2.2 Scoring of complex answers

TREC QA systems have to return up to five ranked strings per question, with each string paired to a document ID. For the evaluation, human assessors judge each 50-byte or 250-byte string that a system returned and decide whether or not it contains an answer to the question in the context of the document. If an incorrect answer is successfully extracted from a document that asserts this incorrect information, it is counted as correct.[15] A system's score for a question is the reciprocal of the rank at which the *first* correct answer is found (or zero if no correct answer is found). This means that "systems are given no credit for retrieving multiple (different) correct answers" (Voorhees 2000). For our first example question about the Costa Rican president, "Calderon" as well as "Figueres" alone have been accepted as correct answers. Conversely, systems are not punished for extracting essentially the same answer several times (from different contexts). In fact, one complex answer extracted from different documents is even forbidden, as each answer has to be accompanied by exactly one document ID.

In our terminology, all separate parts of the following types of complex answers were accepted by TREC assessors: many answers, salience/importance, different measures (pounds or dollars for Concorde fare), different designations (Taiwanese president Li or Lee, Multiple Sclerosis or MS), time dependency, and different beliefs ('Trial by Jury' or 'Thespis' as the first Gilbert and Sullivan opera). In the case of

[15] In TREC-8, a correct answer was counted correct even if the document did not support it. From TREC-9 onwards, these cases are judged 'unsupported'.

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

granularity, it was left to the assessors to decide whether the 'partial answer' was acceptable. Voorhees and Tice (2000) report that this was a major cause of differences among assessors. Collective answers form a special case. Those with three or more items in the answer were excluded when making the test set. The others were reformulated, e.g. from "Who" to "What two US biochemists" (Voorhees and Tice 2000). In this wording, partial answers are not acceptable anymore. Ambiguous questions were also excluded from the test set (the ambiguity between the Congos is newer than the TREC document collection). Thus the problem of complex answer evaluation (are some answers 'better' than others?) was left outside the scope of the past TREC QA tracks.

In the TREC-10 QA track, there will be a separate 'list task' which will feature questions like "Name x countries that import Cuban sugar". The answer should be an *unranked* list of maximally x answer string/document ID pairs. The new QA track guidelines state that "list questions will be scored using instance precision (the fraction of retrieved instances that are correct) and instance recall (the fraction of the target number of instances that are retrieved)". List questions correspond to our 'many answers' type, with the important difference that the required number of items is specified in the list question and systems thus "know" that a complex answer is required. The number specification is necessary for the recall computation in TREC, as they would otherwise have to know the 'true' number of instances.

3.2.3 Systems

In the previous section, we showed that the recognition of complex answers is not rewarded by the TREC scoring rules (with the exception of explicitly asked for collective answers). Therefore, participating groups probably did not make an effort to handle these cases. As an example of how another system produced its answers, we will now briefly describe Textract/QA (Srihari and Li 2000) which was the best performing system at TREC-8 with an MRR of 0.66 for the 50-byte QA task.

The system consists of two components: One for processing the question, and another for processing the documents. A matcher module links the two. The question processor determines the *asking point*, i.e. the kind of 'thing' that is asked for. In many cases, the asking point is a *Named Entity*. For example, "who"-questions ask for a person, "which city"-questions obviously for a city, "what"-questions for an unspecified Named Entity. "Why"-questions do not ask for an entity but for a 'reason'. The question processor also extracts all keywords from the question and expands them with synonyms and morphological variants from a lexicon, e.g. "get/got" are added as synonyms of "win (a prize)". The text processor sends the question to a search engine and tags the top 200 retrieved documents for Named Entities. The matcher module then ranks all sentences that contain a Named Entity matching the question's asking point. Ranking is based on the number of unique keywords in the sentence, their order compared to the order in the question and on whether the verb is identical or a synonym of the question verb. The top five ranked sentences are then trimmed to 50 bytes and returned as results.

In the present context, the most important difference between Shapaqa and

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

Textract/QA is that the former ranks (keyword) answers whereas the latter ranks strings that (are supposed to) contain answers. Therefore, Textract/QA might rank two strings for example first and second that in fact contain the same answer, in different contexts. In principle, Textract/QA and many other TREC systems using similar approaches could also return frequency ordered (keyword) answers. However this would mean performing some normalization of answers/Named Entities (like mapping to heads) and abandoning the current ranking algorithm. As the ranking algorithm is an important part of most systems, this would in fact alter their nature. We think that frequencies are better suited to be presented to users since they are easily understandable whereas scores like 284.40 (Moldovan, Harabagiu, Pa sca, Mihalcea, Goodrum, Gîrju and Rus 2000) used for ranking are totally opaque to users.

No reference to complex answers is made in the description of Textract/QA. In the other TREC-8 systems (which mostly use a similar approach), we find some details that are useful for certain types of complex answer problems. Abney, Collins and Singhal (2000) mention a rule that full dates are preferred if the query contains the words 'day' or 'month', and years-only dates if the query contains 'year'. Thus this rule chooses the appropriate granularity only if it is specifically asked for. Breck, Burger, Ferro, House, Light and Mani (2000) use coreference to find out which phrases in one document actually refer to the same (real world) entity. The entities are then ranked. For answer construction, the *longest realization* of the highest ranked entity is used. This strategy would thus prefer complex answers over simple ones for the granularity type. The modules using entity frequency in Radev, Prager and Samn (2000) and Abney, Collins and Singhal (2000) will be discussed in the next section.

3.3 First solutions

Some of the complex answer types seem to need much pragmatic and world knowlegde to be handled properly, e.g. for automatically resolving ambiguity or finding out which measure or granularity a user would prefer. This is beyond the scope of current systems and even those of the near future. However, in this section we advocate the use of answer frequencies as a first step towards tackling the problem.

3.3.1 Frequencies and context

SHAPAQA's answer sorting based on frequencies, which we implemented to demote parser errors and wrong information to the bottom of the list, proved also useful for complex answer questions. Here are some examples of answers with frequencies that SHAPAQA GR or CT found. Only the top of the list and the first evidence sentence for each keyword answer is shown. We argue that this output format helps users to recognize that a complex answer exists.

- Ambiguity: "What is the capital of Congo?" (CT)
  - 3 x KINSHASA: **Kinshasa** *is* distinctive not only as *the capital of Congo* but ...

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

— 3 x BRAZZAVILLE: ... **Brazzaville** *is the capital of Congo.*

The frequencies and the context given allow users to verify that the two different answers are not due to an error of the QA system. They are thus alerted and can follow the given links to find out more. Note that a system ranking strings that (are supposed to) contain answers would present the user with six strings here, sorted in some way, leaving it to the user to find out that three of them contain "Kinshasa" and three "Brazzaville".

- Time dependency: "Who was Secretary of State during the Nixon administration?" (CT)
  - — 7 x KISSINGER: 11) **Henry Kissinger** was National Security Adviser and *Secretary of State* during the *Nixon administration.*
  - — 4 x ROGERS: By some measures , she wields less influence than any *secretary of state* since the *Nixon administration* , when **William P. Rogers** was ...

  The above information should be enough to convince users that there were two secretaries.
- Different beliefs: "Who killed Kennedy?" (GR)
  - — 9 x AN: ... John Fitzgerald *Kennedy* was *killed* by **an** assassin's bullets as his motorcade wound through Dallas, Texas.
  - — 6 x CONSPIRACY: Instead, they believe that **a conspiracy** *killed Kennedy.*
  - — 5 x OSWALD: If **Oswald** had not *killed Kennedy*, then someone else would have.
  - — 2 x CIA: And explains that **the CIA** *killed Kennedy* because he wanted to tell the American people about the aliens (see: The UFO/JFK Connection
  - — 1 x COMPLEX: such as Mrs. Ventura, he later explained), and that **the military-industrial complex** *killed Kennedy* because he supposedly had decided to withdraw from Vietnam.
  - — 1 x SYNDICATE: held to the flat earth view or to Historian X's thesis that *Kennedy* was *killed* by **a crime syndicate**
  - — 1 x MAFIA: that was accumulated, and how the evidence usually allows ruling out theories that *Kennedy* was *killed* by **the Mafia**, or by the CIA, or by the KGB.
  - — 1 x LONE: "and thus" proves the 1964 Warren Commission conclusion that *Kennedy* was *killed* by **a lone** assassin, Lee Harvey Oswald.
  - — 1 x CESAR: **Cesar** *killed* Robert *Kennedy* from behind while ...
  - — 1 x SPEAR: Qld. Explorer Edmund *Kennedy* was *killed* by **spear**.

  The first and the eighth answer are due to a tagger error on the unknown word 'assassin' (tagged as an adverb). The second to seventh answer however illustrate nicely the different beliefs about Kennedy's death. The context ("they believe", "thesis", "theory that") allows users to identify most answers as beliefs, not facts. The last two answers are due to the ambiguity of 'Kennedy'. In contrast to our Congo example however there is a preferred interpretation of 'Kennedy' as 'John F. Kennedy', and this is clearly reflected by the low answer frequencies belonging to the 'Robert' and 'Edmund' interpretations.

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

- Salience/importance of answers: "What does the Peugeot company manufacture?" (GR)
  - — 2 x VEHICLES: A subsidiary of French auto giant PSA Peugeot Citroën, PCA *manufactures* and imports *Peugeot* and **Citroën vehicles**, ...
  - — 2 x CARS: Sevel Argentina SA Sevel Argentina imports and *manufactures Peugeot* and **Citroën cars**.
  - — 1 x ABSORBERS: (*Peugeot* is the only car company that *manufactures* **its own shock absorbers** so it can get the rates just right.
  - — 1 x TOOLS: Well-known as a maker of bicycles and motor scooters, *Peugeot* also *manufactures* **power tools** and the pepper grinder which the company patented last century.

  Although the frequency differences are tiny, they help to promote the most salient answers (vehicles, cars) to the top of the list. A much larger difference can be seen in the following example: "When did Nixon visit China?" (GR)
  - — 37 x 1972: 7 *Nixon visited China* **in February 1972**, and signed the Shanghai Communiqué declaring historic US-China rapprochement .
  - — 3 x FEBRUARY: President *Nixon visited China* **in February** of 1972, marking the end of the stalemate in bilateral ties.
  - — 1 x 1970: To chose a political event from recent history (*Nixon visited China* **in February 1970**) as the subject was a daring and dangerous act.
  - — 1 x 1989: When Richard *Nixon visited China* **in November 1989**, Deng Xiaoping told him "we can never forget".

  The 1972 visit is historically much more important than the one in 1989. This is clearly reflected by the frequencies. (The 1970 answer is wrong information.)

The MURAX system (Kupiec 1993) answers questions on the basis of an encyclopedia. It performs normalization for potential answers referring to persons by using the title of the article. For instance, in the article "John F. Kennedy" the last name 'Kennedy' is taken to refer to John F. Kennedy. Frequency is taken into account for answer extraction: "Other things being equal, an answer hypothesis having more instances of the match is preferred."

Radev, Prager and Samn (2000) describe a QA system that implements two different algorithms for answer ranking. The algorithm called 'Werlect' uses the frequency of a potential answer Named Entity as one of five features which determine its rank. Frequencies are only counted in the 10 highest ranked passages returned by the passage retrieval component of the system. Answer entities are not normalized to any canonical form (like the head in SHAPAQA) so for example a full name and a last name would be counted as two different entities. In summary, although frequencies are used, their status in the system is less influential than in SHAPAQA.

In the system described by Abney, Collins and Singhal (2000) some normalization of potential answer Named Entities is performed. Dates are mapped to a year-month-day format and person names to their last word (their head in our terminology, frequently the last name). Frequencies are then counted in the 50 highest ranked passages from the passage retrieval component of the system and are used for ranking all entities which match the entity type that is asked for in the question.

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

As a special rule, occurrence in the passage(s) "that received the maximal score from the passage retrieval component" counts ten times as much as occurrence in any lower ranked passage. As complex answers were no issue in the past TREC QA evaluations, Abney, Collins, and Singhal do not discuss the impact of their frequency approach on them.

3.3.2 Complex answers in one sentence

The previous section showed how answer frequency and context can help users to reconstruct a complex answer. However we can actually do better than this for some cases of complex answers.

- Collective answer: "What two researchers discovered the double-helix structure of DNA in 1953?" (CT)
  - — 8 x WATSON: In *1953*, **James Watson** and Francis Crick *discovered the double-helix structure of DNA*.
  - — 4 x CRICK: In *1953*, James Watson and **Francis Crick** *discovered the double-helix structure of DNA*.
  - — 2 x PRESIDENT: among others, for its Watson School of Biological Sciences **whose current president**, James D Watson , *discovered the double-helix structure of DNA* in *1953*.

  At the moment, the answers 'Watson' and 'Crick' are treated separately. However, the QA system could easily check that the first evidence sentence for 'Watson' also contains 'Crick' (in fact it is identical to the first sentence for 'Crick'). It could then output a combined answer like "8 x WATSON, 4 x CRICK: In *1953* , **James Watson** and **Francis Crick** *discovered the double-helix structure of DNA*." This is an especially efficient strategy if for some reason (e.g. participation in TREC) the system has a limited space for answers.
- Granularity: "Where was Lincoln assassinated?" (CT)
  - — 77 x FORD: On April 14, 1865, as he sat **in Ford**'s Theater watching a comedy, President *Lincoln* was *assassinated*.
  - — 45 x WASHINGTON: Died: On Good Friday, April 14, 1865, *Lincoln* was *assassinated* at Ford's Theatre **in Washington** by John Wilkes Booth, an actor, who.
  - — 11 x OFFICE: Civil War, and later became the Secretary of War under President James Garfield (who strangely enough was *assassinated* **in office** just like Abraham *Lincoln*).
  - — 10 x THEATER: After *Lincoln* was *assassinated* **in this very theater**, he was carried across the street.
  - — 6 x THEATRE: Abraham *Lincoln* was *assassinated* on April 14, 1865 by John Wilkes Booth **at Fords Theatre** and died the next day.

  The answer 'Ford' (from not treating 'Ford's Theater' as one chunk) was found in 77 evidence sentences. As mentioned in section 2, the evidence sentences under the same keyword are not sorted at the moment. The problem of complex answers however suggests a simple new strategy: Prefer the sentence that

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

contains the most other answers. In case of a tie, prefer the higher frequency ones. In the present example, the first evidence sentence for 'Washington', which is also among the 77 sentences for 'Ford' contains the three answers 'Ford', 'Washington' and 'Theatre'. There is even another sentence containing the slightly more frequent variant 'Theater': "... on April 14 , 1865 – one year after the Civil War had ended – Lincoln was assassinated at Ford's Theater in Washington." If this sentence is chosen as first evidence sentence, and all three answers are marked (in bold), the system can in fact prefer a complex answer over a simple one without having to know anything about the underlying semantic reasons for the complexity.

- Different designations: "What is the brightest star visible from Earth?" (CT)
  - — 8 x SIRIUS: Sirius is the brightest star visible from Earth.
  - — 3 x STAR: Sirius, the Dog Star, is the brightest star visible from Earth, exclusive of our Sun, of course.

  The same argument as above holds in this case. Of the eight sentences for 'Sirius', the system would prefer the one containing also '(Dog) Star', and is thus able to prefer a complex answer.

Unfortunately, SHAPAQA did not find any answers of the 'different measures' or 'many answers' types, so we cannot give any examples here. Still, we can assume that the most common measure (in English texts on the Internet) for the quantity at hand will be promoted to the top of the list through frequency sorting. There also is a chance of finding several (probably two) measures in one sentence, which can then be preferred. The same holds for the many answers.

The 'Werlect' algorithm (Radev, Prager and Samn 2000), mentioned in the last section, implements a similar heuristic to the one just proposed. After all potential answer Named Entities are ranked, all 50 (or 250) byte text segments of the passages from the passage retrieval component are ranked using as features the original passage rank, the number of matching words from the query and the rank of all entities contained in the segment. All other things being equal, a segment that contains more higher ranked entities would thus be preferred. Radev, Prager and Samn note that in TREC-8, the 'Werlect' algorithm performed slightly worse than the alternative 'Ansel' algorithm that does not take entity frequencies or multiple entities in one segment into account. We think that there are three possible reasons. First, frequency counts need some normalization of entities. Secondly, preferring segments which contain as many entities as possible while being restricted to a fixed byte length might cause answer strings to contain several potential answers without indicating which one is 'meant'. According to the TREC QA evaluation rules, this answer string has to be judged incorrect. In a realistic QA system, answers strings would probably not have to be truncated. Third, as TREC systems receive no credit for retrieving complex answers, this advantage of the 'Werlect' approach is not rewarded. An example of a TREC-9 system that also counts answer frequencies and the number of potential answers in a segment is Clarke, Cormack, Kisman and Lynam (1999).

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

4 Possible improvements and conclusion

The success of the approaches to complex answer handling that were proposed in the previous section hinges upon the reliability of the frequency counts. Good frequency counts need a method for deciding whether two answers found in two documents refer to the same entity. At the moment this is done by mapping the answers to their heads. It is clear however that this heuristic can be improved upon. It does not work for cardinality questions, as e.g. "16 moons" and "4 moons" would both be mapped to MOONS. In one other case it incorrectly merged two answers ("Mississippi River" and "Missouri River") into one (RIVER). In general two phrases having the same head need not refer to the same entity. In a QA context, however, it is unlikely that e.g. two different persons with the same last name are found as answers. In the case of two different dates in the same year it is reasonable to count both as evidence for the year answer. A bigger problem with the head mapping heuristic is that some answers referring to the same entity are not merged. One such example is the Costa Rican president "José María Figueres Olsen" who was also found as "Jose Maria Figueras". Another example is the 'cars' and 'vehicles' answers for the Peugeot question which would need ontological knowledge to be merged. This is a topic for future research.

In this article we introduced the concept of complex answers. With complex answers, evaluation is no longer a decision about 'correct' versus 'incorrect', as a simple answer may be correct but a complex one 'better'. We identified nine different types of complex answers and proposed two approaches based on answer frequencies that enable QA systems to handle these types without having to understand the semantic reasons for the complexity. As it is not clear at the moment how to evaluate with complex answers, we could only demonstrate the advantages of the approaches using illustrative examples gathered by a WWW QA system. We also described this system which uses shallow parsing modules, especially a grammatical relation finder, based on Memory-Based Learning.

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

Abney, S., Collins, M. and Singhal, A. (2000) Answer extraction. *Proceedings 6th ANLP*. ACL.

Bies, A., Ferguson, M., Katz, K. and MacIntyre, R. (1995) *Bracketing Guidelines for Treebank II Style, Penn Treebank Project*. University of Pennsylvania, Philadelphia.

Breck, E., Burger, J., Ferro, L., House, D., Light, M. and Mani, I. (2000) A sys called qanda. *Proceedings of TREC-8*. pp. 499–506.

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press

Buchholz, S., Veenstra, J. and Daelemans, W. (1999) Cascaded grammatical relation assignment. In: Fung, P. and Zhou, J., editors, *Proceedings EMNLP/VLC-99*. ACL, pp. 239–246.

Clarke, C. L. A., Cormack, G. V., Kisman, D. I. E. and Lynam, T. R. (2001) Question answering by passage selection (mlutitext experiments for trec-9). *Proceedings of TREC-9*.

Collins, M. (1997) Three generative, lexicalised models for statistical parsing. *Proceedings 35th ACL and 8th EACL*. Madrid, Spain.

Daelemans, W., Van den Bosch, A. and Weijters, A. (1997) IGTree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review* **11**: 407–423.

Daelemans, W., Van den Bosch, A. and Zavrel, J. (1999) Forgetting exceptions is harmful in language learning. *Machine Learning*, Special issue on NLE, **34**: 11–41.

Daelemans, W., Zavrel, J., Berck, P. and Gillis, S. (1996) MBT: A memory-based part of speech tagger generator. In: Ejerhed, E. and Dagan, I., editors, *Proc. 4th Workshop on Very Large Corpora*. ACL SIGDAT, pp. 14–27.

Daelemans, W., Zavrel, J., van der Sloot, K. and van den Bosch, A. (2000) *TiMBL: Tilburg memory based learner, version 3.0, reference guide*. ILK Technical Report 00-01, Tilburg University. (Available from `http://ilk.kub.nl`.)

Kupiec, J. (1993) MURAX: A robust linguistic approach for question answering using an on-line encyclopedia. *Proceedings ACM SIGIR'93*. pp. 181–190.

Marcus, M., Santorini, B. and Marcinkiewicz, M. A. (1993) Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics* **19**(2): 313–330.

Moldovan, D., Harabagiu, S., Pa sca, M., Mihalcea, R., Goodrum, R., Gîrju, R. and Rus, V. (2000) Lasso: A tool for surfing the answer net. *Proceedings of TREC-8*. pp. 175–183.

Radev, D. R., Prager, J. and Samn, V. (2000) Ranking suspected answers to natural language questions using predictive annotation. *Proceedings 6th ANLP*. ACL.

Srihari, R. and Li, W. (2000) Information extraction supported question answering. *Proceedings TREC-8*. pp. 185–196.

Voorhees, E. M. (2000) The TREC-8 question answering track report. *Proceedings TREC-8*. pp. 77–82.

Voorhees, E. M. and Tice, D. M. (2000) The TREC-8 question answering track evaluation. *Proceedings TREC-8*. pp. 83–106.

Voorhees, E. M. and Harman, D. K., editors. (2000) *Proceedings of TREC-8*. No. 500-246, NIST Special Publication. (Available at `http://trec.nist.gov/pubs.html`.)

https://doi.org/10.1017/S1351324901002777 Published online by Cambridge University Press