Chapter 4

Answering Definitional Questions

A Hybrid Approach

Sasha Blair-Goldensohn, Kathleen R. McKeown, & Andrew Hazen Schlaikjer

4.1 Introduction

In recent years, question answering (QA) systems such as those in the yearly TREC conferences have reached a remarkably high level of performance. These systems are premised on the *short-answer model,* in which the goal is to answer questions for which the correct response is a number, short phrase, or sentence fragment.

However, many questions that occur in real-life tasks are not in this class. Consider a student asked to prepare a report on the Hajj, an Islamic religious duty. In the context of short-answer QA, both patience and prescience will be required to elicit the core facts. First, a relatively long list of questions would be required (e.g., "Where is the Hajj carried out?" "How long does it last?" "Who undertakes a Hajj?" etc.). Second, knowing which questions to ask requires knowledge that the questioner likely does not have. That is, the questions that best elicit a description of one thing (e.g., the Hajj) can be quite different than those best suited for finding out about something else (e.g., the Caspian Sea).

Producing rich, multi-sentence responses to open-ended questions—such as those requiring definitions, biographies, or opinions as answers—is the focus of *long-answer* QA. This area is still in early stages of development, but already the subject of several pilot studies and much active research (Voorhees 2003). In this chapter, we will concentrate on *definitional* QA, the task of providing long answers to "What is X?" type questions.

Definitional QA systems are not only interesting as a research challenge. They also have the potential to be a valuable complement to static knowledge sources like encyclopedias. This is because they create definitions dynamically, and thus answer definitional questions about terms which are new or emerging. They also can tailor an answer to a user's needs, for instance by cre-

ating a longer or shorter definition, or one which is drawn from specified knowledge sources.

We will focus our discussion around DefScriber (Blair-Goldensohn et al. 2003), a definitional QA system that implements a hybrid of goal-driven (knowledge-based) and data-driven (statistical) methods. The *goal-driven techniques* (section 4.4) shape the answer in a top-down manner, using a set of definitional predicates modeled on typical elements of a definition. *Data-driven techniques* (section 4.5) operate in a bottom-up way, using statistical measures to identify themes in the data. An overview of DefScriber's operation is given in section 4.3. We conclude the chapter by presenting results of a recent evaluation which show DefScriber's hybrid approach achieving significant improvement over a competitive baseline.

4.2 Related Work

Our hybrid approach to definitional QA builds on research in summarization and generation. Previous work in multi-document summarization has developed solutions that identify similarities across documents as the basis for summary content (Mani and Bloedorn 1997, Carbonell and Goldstein 1998, Hatzivassiloglou et al. 1999, Barzilay et al. 1999, Radev et al. 2000, Lin and Hovy 2002). (The use of summarization techniques in the context of short-answer QA is discussed in chapter 17.) Whether similarities are included through sentence extraction or information fusion (Barzilay et al. 1999), all of these approaches are data-driven because similarities in the data determine content.

Goal-driven, or top-down, approaches are more often found in generation. Schemas (McKeown 1985), rhetorical structure theory (Mann and Thompson 1988, Moore and Paris 1992, Hovy 1993, Marcu 1997) and plan-based approaches (Reiter and Dale 2000) are examples of goal-driven approaches, where the schema or plan specifies the kind of information to include in a generated text. In early work, schemas were used to generate definitions (McKeown 1985), but the information for the definitional text was found in a knowledge base. In more recent work, information extraction is used to create a top-down approach to summarization (Radev and McKeown 1998) by searching for specific types of information which can be extracted from the input texts (e.g., perpetrator in a news article on terrorism). Here, the summary briefs the user on domain-specific information assumed a priori to be of interest.

Other long-answer QA systems are currently under development as part of the AQUAINT program (Voorhees 2003). Some of these share attributes with DefScriber. In chapter 5, Weischedel et al. explores biographical questions using a combination of methods that are largely complementary to those used in DefScriber—namely, identification of key linguistic constructions and information extraction (IE) to identify specific types of semantic data.

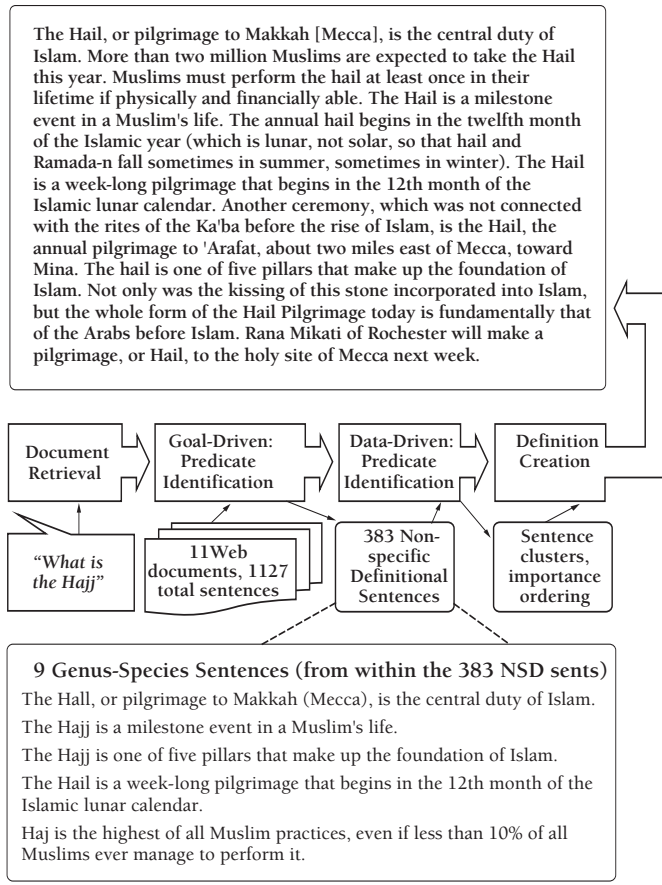

Figure 4.1. DefScriber's Operation on the Question "What is the Hajj?"

Another important contrast between DefScriber and most of the long-answer systems developed under the AQUAINT program has to do with answer format. While these systems (including the one presented in chapter 5) mostly produce answers as a ranked list of descriptive phrases or sentences, DefScriber uses summarization methods to produce a coherent, multi-sentence, encyclopedia-style definition.

4.3 DefScriber: Architecture Overview

Before turning to the development of our goal- and data-driven methods, a brief overview of DefScriber is useful to understanding where these pieces fit. Figure 4.1 gives a high-level view of DefScriber's operation, illustrating input

Predicate	Description	Instance Example
Non-specific Definitional	Any type of information relevant in a detailed definition of the term. NSD are a superset of the below predicates.	Costs: Pilgrims pay substantial tariffs to the occupiers of Makkah and the rulers of …
Genus	Category to which term belongs.	The hajj is a type of ritual.
Species	Describes properties other than or in addition to Genus. Species are a superset of the below predicates.	The annual hajj begins in the twelfth month of the Islamic year.
Target Partition	Divides the term into two or more conceptual or physical parts.	Qiran, Tammatu' and Ifrad are three different types of Hajj.
Cause (effect)	States explicitly that the term is the cause (effect) of something.	The pilgrimage causes the past sins of a Muslim to be forgiven.
History	Gives historical information relating to the term.	Mohammed, founder of Islam, started the tradition in 632 C.E.
Etymology	Information on the term's genesis, e.g., adaptation from another language.	In Arabic, the word Hajj means a resolve of magnificent duty.

Table 4.1. Definitional Predicates: Descriptions and Examples

and output of each stage. This example traces an actual answer generated for the question "What is the Hajj?" during our evaluation (section 4.6).

The input is specified as a question, which feeds into the document retrieval phase. The user can also specify which databases to search, a maximum number of documents to retrieve, and the desired answer length.[1]

The information retrieval (IR) module uses a fixed set of patterns to identify the term to be defined in the question, and then generates a set of search queries in order of decreasing expected precision with respect to that term. Queries are sent to a search engine until a threshold number of documents has been retrieved.[2]

Once documents are retrieved, the primary goal-driven step is performed, with the system examining documents for instances of definitional predicates. Next, data-driven analysis is performed to produce sentence clustering and ordering information. In the last step, a definitional answer is created via sentence extraction, guided by the analysis done in the goal- and data-driven stages.

4.4 Definitional Predicates: A Goal-Driven Approach

Answering a "What is X?" definitional question and creating a summary of query results for the search term x are strongly related problems. Yet, as readers, we have more specific expectations for a definition than for a general-use summary. The idea of definitional predicates is to model these special properties of a definition so the system can use them to create better answers.

4.4.1 The Predicate Set

Our set of definitional predicates is shown in table 4.1. Currently, the system automatically identifies instances of three of these types in text: genus, species and nonspecific definitional (NSD). Active research on identifying target partition and history instances is ongoing.

An important distinction is that NSD subsumes all of the other more specific predicate types that appear underneath it in table 4.1. Thus, identifying NSD text is crucial because it is a cue to the presence of other predicates; it also removes noise and provides a set of useful definitional text which is given as input to data-driven methods even when the text cannot be further classified with a more specific predicate. We choose genus and species as the first specific predicates to implement because they are at the core of what definitions are. Related work (Sager and L'Homme 1994, Swartz 1997, Sarner and Carberry 1988) consistently identifies these two concepts as key parts of defining a term.

We build on work such as (Klavans and Muresan 2000), who acquire dictionary-type definitions, and (Sarner and Carberry 1988), who propose three "strategic predicates," including identification and properties predicates which are analogous to our genus and species, respectively. Research in terminological theory (Sager and L'Homme 1994) and philosophy (Swartz 1997) also theorizes that the type of information modeled by many of our predicates (including genus, species, synonym and target partition) is crucial to descriptive-type definitions.

Our definitional predicates contrast with those from rhetorical structure theory (RST) (Mann and Thompson 1988, Moore and Paris 1992, Hovy 1993, Marcu 1997) in that our predicates do not represent the intent or overall structure of a document in their definition. That is, each unit of text in a source document can be evaluated intrinsically as to whether it instantiates a definitional predicate.

To use these predicates in our system, we must identify units of text which contain them. To do this, we first did a manual examination of documents to create sample data annotated with predicates. Using this data, we explored two approaches to identifying predicates. The first uses machine learning to learn a feature-based classification that predicts when a predicate occurs. The second uses pattern recognition over patterns extracted from the annotated data.

4.4.2 Creating a Training Set

To produce the training data for DefScriber, coders marked 81 total documents for instances of the predicates in table 4.1. The data included 55 documents marked by one coder and 13 marked by two coders. To gather documents, 14 terms were first selected for broad coverage from several diverse categories: geopolitical, science, health, and miscellaneous. Then we retrieved approximately 5 web documents for each term using a process similar to our system's IR component.

4.4.3 Rule Extraction: Statistical Techniques

Using this set of annotated documents, we applied machine learning techniques and tools to extract rules for predicate identification. These approaches allow us to efficiently discover relationships between text features and the presence of definitional predicate instances. The text unit we use here is the sentence; that is, we wish to discover rules that will take the features of a source sentence as input, and output those (if any) definitional predicates that are predicted to be in the sentence.[3]

Feature selection was done in part using observations from document markup. For instance, we include several features measuring a sentence's "term concentration," i.e., the term's frequency within a sentence and/or nearby sentences, based on the observation that appearance of the term appears to be a predictor of definitional material. We also include features for relative and absolute position of a sentence in a document, based on the observation that key information tends to concentrate toward the top of documents. Other features, such as presence of punctuation, are added to detect full-sentence text (as opposed to headings or other fragments), since most predicates other than NSD seem to occur mainly in full sentences. Some "blind" features such as bag-of-words are also used.

We applied two machine learning tools to the learning problem: the rule-learning tool Ripper (Cohen 1995) and the boosting-based categorization system BoosTexter (Schapire and Singer 2000). Both algorithms performed similarly in terms of the accuracy of their predictions on test data; Ripper's rules are used in DefScriber since they were somewhat simpler to implement.

The nonspecific definitional (NSD) predicate, which indicates a sentence's relevance to any aspect of defining the term, fares well using rules that consider term concentration and position in document. Using cross-validation, accuracy of 81 percent was obtained with Ripper (76 percent using BoosTexter). This is sufficient for DefScriber since this predicate is not used to place sentences directly into the definition, but rather to pare down noisy and voluminous input by pulling out sentences which merit further examination.

4.4.3 Rule Extraction: Syntactic and Lexical Patterns

Using lexicosyntactic patterns manually extracted from the predicate-annotated documents, we create a set of high-precision patterns for the two predicates most core to definitions: genus and species. We currently model sentences containing both genus and species information, as these "G-S" sentences provide a strong grounding context for understanding the term. G-S sentences situate a term in a higher level category (its "genus"), and the "species" information in such sentences tends to give key features that distinguish a term within that category.

Rather than modeling the patterns at the word level, i.e., as flat templates with slots to fill, we model them as partially specified syntax trees (figure 4.2).

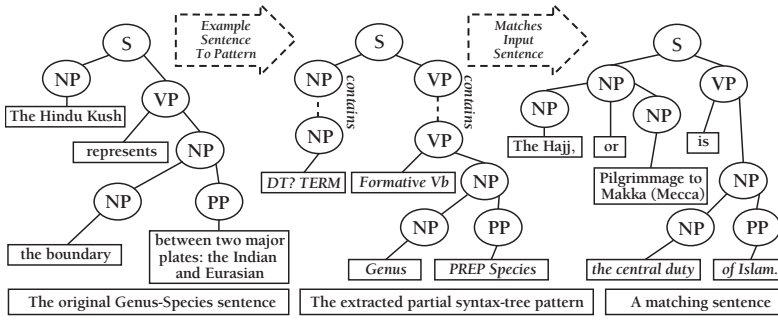

Figure 4.2. Pattern Extraction and Matching for a Genus-Species Sentence.

One such pattern can match a large class of semantically similar sentences without having to model every type of possible lexical variation. This approach derives from techniques used in information extraction (Grishman 1997), where partial *sub*trees for matching domain-specific concepts and named entities are used because automatic derivation of full parse trees is not always reliable. However, data-driven techniques (section 4.5) offer additional protection from false or extraneous matches by lowering the importance ranking of information not corroborated elsewhere in the data.[4]

Figure 4.2 illustrates the transformation from example sentence to pattern, and then shows a matching sentence. Our patterns are flexible—note that the example and matched sentences have somewhat different trees. Another point of flexibility is the verb itself; FormativeVb will match various forms of verbs in a set which our matching algorithm considers expressive of "belonging" to a category (e.g., "be," "exemplify").

Using our predicate-annotated data set, we manually extracted 18 distinct patterns which match G-S sentences. These 18 patterns provide sufficient recall to reliably find at least one instance in modestly sized document sets; over our evaluation test set (see section 4.6), at least one G-S sentence was identified for 16 of 19 terms, with a mean of 3.5 G-S sentences per term (culled from a mean of 15 documents retrieved). Precision was 96%, recall unknown.

4.5 Data-Driven Techniques: Applying Summarization

While our set of predicates, including genus and species, are domain-neutral, they are not meant to model all possible important information for a given term definition. Some information types may be hard to define computationally a priori. Also, a given sentence may instantiate a definitional predicate but include only peripheral content. We address these issues in the data-driven stage of DefScriber's pipeline (refer to figure 4.1), applying statistical tech-

Source	Term
Pilot Evaluation	asceticism, Aum Shinrikyo, battery, fibromyalgia, *gluons*, goth, Hajj, Mobilization for Global Justice, nanoparticles, religious right, *Shining Path*, Yahoo!
Hand-picked	autism, Booker Prize, Caspian Sea, East Timor, *hemophilia*, *MIRV*, orchid, pancreas, passive sonar, skin cancer, tachyons, *tsunami*

Table 4.2. Evaluation Terms (in Italics Were Used for Training, the Rest for Testing)

niques adapted from multi-document summarization to the nonspecific definitional sentences identified in the goal-driven stage.

First, a *definition centroid* is computed by creating a stemmed-word vector of all the NSD sentences. Then the individual sentences are sorted in order of decreasing "centrality," as approximated by IDF-weighted cosine distance from the definition centroid. This method creates a definition of length *N* by taking the first *N* unique sentences out of this sorted order, and serves as the TopN baseline method in our evaluation. Note that this method approximates centroid-based summarization, a competitive summarization technique (Radev et al. 2000).

After ordering sentences with TopN, we perform a non-hierarchical clustering that we use to decrease redundancy by avoiding same-cluster sentences in the answer. Since our clustering similarity measure uses IDF computed over a large collection, it can suffer from overweighting of specialized terms; to account for this, we augment the cosine distance calculation, using local IDF values calculated dynamically from the pool of NSD sentences.

The final data-driven technique improves cohesion by considering the content of the previous answer sentence when choosing a sentence to add to the answer. After choosing the first sentence as in TopN, we choose each remaining sentence as the top-ranking sentence from the cluster that minimizes cosine distance from the definition centroid and the previously chosen sentence's cluster.

DefScriber's default configuration integrates all the above data-driven techniques—TopN, clustering, local IDF weighting, and cohesion ordering—combining them with the goal-driven method of genus-species predicate identification. We place the top-ranking (in terms of TopN) G-S sentence first in the definition, and use the cohesion-based ordering to add the remaining sentences. We call this integrated goal- and data-driven method *DefScriber.*

4.6 Evaluation

Our evaluation used human judgments to measure the performance of DefScriber's definitions over a set of varied terms. By surveying users on definitions generated by different configurations of DefScriber, we are able to measure the improvement of DefScriber over the baseline (TopN) method. We address five main qualities in our survey: relevance (precision), redundancy, structure, breadth of coverage, and term understanding.

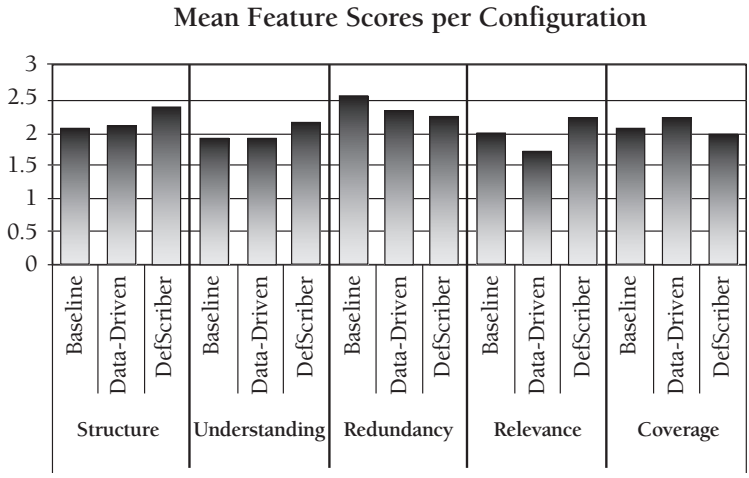

Figure 4.3. Evaluation Results

We chose a set of 24 terms5 (table 4.2) for which to create answer definitions using the Internet as our knowledge source. We picked half of the terms ourselves, aiming for varied domain coverage; the other half were randomly chosen from among the definitional questions in the AQUAINT pilot evaluation (Voorhees 2003). For each of the test terms, we evaluate three different system configurations: baseline (TopN only), data-driven (all data-driven methods from section 4.5), and full DefScriber (data- plus goal-sriven; described at end of section 4.5). Each answer was ten sentences long.

38 judges participated in the evaluation, and each was asked to rate definitions for 6 different test terms. This resulted in an average of four rated samples for each of the 57 answer definitions (19 test terms, three system configurations). Figure 4-3 shows the resulting mean feature scores for each system configuration.

DefScriber achieves the best scores in structure, redundancy, term understanding, and relevance, with statistically significant margins in the first two categories ($P < .10$) using RIDIT analysis.[6] In coverage, data-driven does best, and DefScriber worst, but none of the differences are significant.

With the best performance in four of five categories, DefScriber is clearly the best configuration. The leading genus-species sentence significantly improves the answer definition's structure by giving readers solid orientation and context. DefScriber's top score in term understanding, while not statistically superior, suggests that this context helps readers to understand the more detailed information that may follow.

4.7 Future Work

Future work on DefScriber will concentrate on increasing the number of defi-

nitional predicates automatically identified by the system, as well as on improving identification performance on such predicates.

We are currently working to improve our feature-based predicate identification methods by growing our annotated data set while also extracting more and richer features to input into our machine learning methods. To improve the pattern-based methods, we are actively working with IE bootstrapping techniques developed in Snowball (Agichtein and Gravano 2000) to automatically learn predicate patterns from manually extracted "seed" examples. Such techniques would allow us to supplement our manually-generated patterns and bring new predicates online more quickly.

4.8 Conclusion

Definitional question answering is an emerging research area that goes beyond traditional short-answer questions. In this chapter, we examined the approach of DefScriber, a system that answers definitional questions with dynamically created paragraph-style responses, and uses a combination of goal-driven and data-driven methods to make its definitions fluent and informative. These methods include goal-driven techniques that screen relevant material from voluminous search results, and identify sentences containing types of information typically used in definitions. DefScriber complements these techniques with a set of data-driven methods which guide answer content in a bottom-up manner, using statistical analyses of the data. Our evaluation results demonstrate that the system significantly outperforms competitive summarization baselines. While our ongoing work will continue to improve the system, these results show the promise of the DefScriber's hybrid approach for definitional question answering.

Acknowledgments

The authors wish to acknowledge the support of ARDA through AQUAINT contract MDA908-02-C-0008. We are also thankful for the generous and thoughtful contributions of colleagues at Columbia University and at the University of Colorado-Boulder.

Notes

1. Currently, DefScriber can run its search against the Internet as well as several local document collections. Although we currently consider the web as a set of flat documents, chapter 17 considers more rich representations of web pages for QA.

2. The current system recognizes only questions of the form, "What is/are a/the ...," and queries generated for IR are similarly limited. As question recognition and query generation are not an innovative aspect of DefScriber, they are not discussed further in this chapter.

3. This prevents the learning of rules for a predicate instance that spans sentences. However, with our current predicate taxonomy, our document markup found such instances exceedingly rare.

4. For instance, in our "Hajj" example, the system matches the G-S sentence: "The Hajj was Muhammad's compromise with Arabian Paganism." This sentence is in principle a correct match, but the genus and species given here are extraneous and metaphorical. The fact that this information is less central to the definition will be detected statistically by our data-driven methods (section 4.5), and the sentence will thus be ranked below more central G-S sentences.

5. DefScriber is a robust system that produces a definition for virtually any term contained in the document collection; we limited the size of our evaluation for practical purposes.

6. Since the rating scales used by judges are ordered metrics, we analyze the results with RIDIT (Fleiss 1981) analysis, a technique that accounts for the natural ordering information in these measures, i.e., the fact that poor and so-so are both below good, not simply separate categories.

Sasha Blair-Goldensohn is a PhD student in computer science at Columbia University. His research interests include question answering and text summarization. He can be reached via www.cs.columbia.edu/~sashabg.

Kathleen R. McKeown (PhD, University of Pennsylvania) has been a professor of computer science at Columbia University since 1982, serving as department chair from 1997 to 2003. Her interests include summarization,natural language generation, multimedia explanation, and digital libraries.

Andrew H. Schlaikjer is a researcher and programmer in Columbia University's Natural Language Processing Group, focusing on question answering systems. He can be reached at hazen@cs.columbia.edu.