Instance-Based Question Answering

Lucian Vlad Lita

CMU-CS-06-179

December 2006

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Jaime Carbonell, Chair
Eric Nyberg
Tom Mitchell
Nanda Kambhatla, IBM TJ Watson

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

This research was sponsored by the Department of Interior under contract no. NBCHC040164, the Department of Defense under contract no. MDA908-02-C-0009, and the Defense Advanced Research Projects Agency (DARPA) under SRI International subcontract no. SRI 000691.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: statistical question answering, QA, natural language processing, statistical answer extraction, question clustering, answer type distributions, cluster-based query expansion, learning answering strategies, machine learning in NLP

To my wife Monica

Abstract

During recent years, question answering (QA) has grown from simple passage retrieval and information extraction to very complex approaches that incorporate deep question and document analysis, reasoning, planning, and sophisticated uses of knowledge resources. Most existing QA systems combine rule-based, knowledge-based and statistical components, and are highly optimized for a particular style of questions in a given language. Typical question answering approaches depend on specific ontologies, resources, processing tools, document sources, and very often rely on expert knowledge and rule-based components. Furthermore, such systems are very difficult to re-train and optimize for different domains and languages, requiring considerable time and human effort.

We present a fully statistical, data-driven, **instance-based** approach to question answering (IBQA) that learns how to answer new questions from similar training questions and their known correct answers. We represent training questions as points in a multi-dimensional space and cluster them according to different granularity, scatter, and similarity metrics. From each individual cluster we automatically learn an answering strategy for finding answers to questions. When answering a new question that is covered by several clusters, multiple answering strategies are simultaneously employed. The resulting answer confidence combines elements such as each strategy's estimated probability of success, cluster similarity to the new question, cluster size, and cluster granularity. The IBQA approach obtains good performance on factoid and definitional questions, comparable to the performance of top systems participating in official question answering evaluations.

Each answering strategy is cluster-specific and consists of an expected answer model, a query content model, and an answer extraction model. The expected answer model is derived from all training questions in its cluster and takes the form of a distribution over all possible answer types. The query content model for document retrieval is constructed using content from queries that are successful on training questions in that cluster. Finally, we train cluster-specific answer extractors on training data and use them to find answers to new questions.

The IBQA approach is resource non-intensive, but can easily be extended to incorporate knowledge resources or rule-based components. Since it does not rely on hand-written rules, expert knowledge, and manually tuned parameters, it is less dependent on a particular language or domain, allowing for fast re-training with minimum human effort. Under limited data, our implementation of an IBQA system achieves good performance, improves with additional training instances, and is easily trainable and adaptable to new types of data. The IBQA approach provides a principled, robust, and easy to implement base system which constitutes a robust and well performing platform for further domain-specific adaptation.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

In a time many refer to as the *information age* people are indeed surrounded by overwhelming quantities of information. One of the main problems addressed in current research is the need for efficient and effective methods of accessing information. Very often professional data analysts and private users have specific questions that require specific answers. These answers are typically hidden in vast amounts of data collections, and users need focused, confident information from trusted sources. In recent years, the field of question answering has started to address this problem. Before question answering, researchers had considered information need of a different granularity (e.g. documents instead of answers) or had devised extraction techniques tailored to specific domains.

The field of *Information Retrieval* (IR) has focused on retrieving relevant documents and passages from very large text corpora using statistical methods. While this focus is a perfect match for a variety of tasks, very often a user's information need is more specific and browsing complete documents for answers to questions is slow and far from optimal.

Moreover, IR is generally not concerned with understanding the meaning of queries when posed in natural language – e.g. in the form of a question.

Information Extraction (IE) overcomes the specificity problem by attempting to extract very specific nuggets of information (e.g. names of companies, their role in transactions, their partners etc) from text. It also has the advantage of being easily applied to large text corpora. However, the information nuggets are extracted according to pre-defined templates (e.g actor-action-role) and/or pre-specified topics (e.g. business mergers, terrorist activity etc). Because they are highly specialized, information extraction templates are domain dependent and are not easily portable.

Question Answering (QA) is one of the more recent tools researchers are developing in order to obtain efficient and effective access to data for specific information requests. Very often the information required by a user or analyst is contained in a paragraph, sentence, or phrase. The field of question answering addresses this problem by attempting to find focused, exact answers to natural language questions from large collections of text.

The *Text REtrieval Conference* (TREC[1]) is a series of workshops initiated in 1992 that facilitate exchange of research ideas on text retrieval methods for various tasks (document retrieval, question answering, genomics domain document retrieval, novelty track etc) as well as an annual evaluation of multiple systems for each individual track. The question answering track (TREC QA track) [122, 123, 124, 125] is one of the task evaluations that has been established in 1999 (TREC-8). Each year systems are provided with a large local collection of documents and approximately 500 unseen questions to be answered over the period of a week without human intervention.

Most questions in the TREC evaluation are open-domain and expect short, factual answers. These types of questions are often called *factoid questions*. One of the advances prompted by TREC is a more standardized evaluation for question answering. Although still problematic, evaluating answer correctness can be done using answer patterns – i.e. regular expressions constructed from known correct answers – or by pooling answers from all

[1] TREC is co-sponsored by the National Institute of Standards and Technology (NIST), Information Technology Laboratory's (ITL) Retrieval Group of the Information Access Division (IAD), and by the Advanced Research and Development Activity (ARDA) of the U.S. Department of Defense.

participating systems and then using human assessors to evaluate answer correctness.

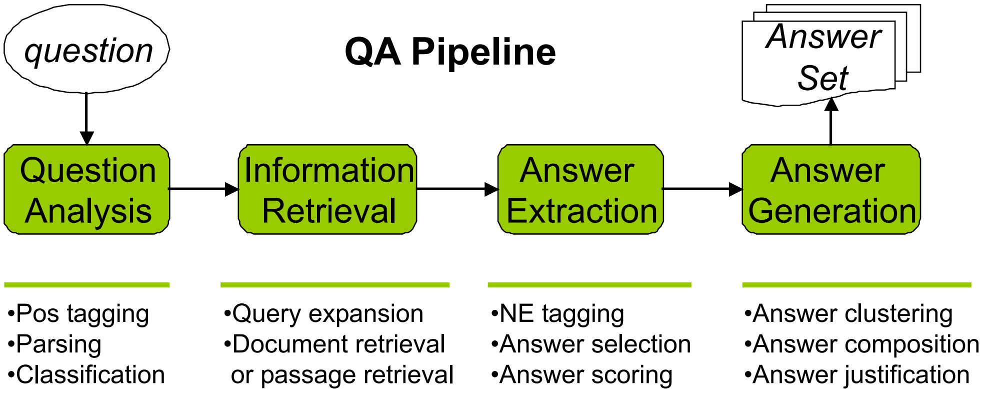

Figure 1.1: Stages of a question answering pipeline system. Most systems follow to some extent the same question answering pipeline structure: question analysis, information retrieval, answer extraction/selection, and answer generation. In the Question Analysis stage includes *answer modeling*: finding the structure and form of the expected answer - most often done through answer type classification.

Researchers have followed many directions in question answering including: question parsing [51, 86] and classification [16, 51, 86, 56, 136], using available resources such as WordNet [51, 97, 99], extracting answers from Web documents [16, 62], statistical approaches to answer extraction and selection [32], semantic analysis [50, 131, 86], reasoning and inferencing [86], knowledge intensive question answering [46], flexible QA system architectures [94], answering complex questions [110, 41], information extraction centric QA [112, 2, 111, 105], and cross lingual QA systems [75, 76].

Most question answering research has at its core a standard *pipeline* QA system [87, 98, 49, 21] that combines several components in a sequential fashion 1.1. Such question answering systems include components corresponding to the following stages in the question answering process:

1. *question analysis* – the stage in which questions are processed (e.g. part of speech tagging, named entity extraction, parsing), analyzed, and classified according to various ontologies. Answer type classificationis a specific method of answer modeling, through which the QA system attempts to identify the structure and type expected

answer.

2. *information retrieval* – the stage in which queries are formulated according to query types, question keywords, and additional content. Based on these queries, relevant documents or passages likely to contain correct answers are retrieved.

3. *answer extraction* – the stage in which candidate answers are extracted from relevant documents and assigned a confidence score – i.e. the extractor confidence that the candidate answer is correct.

4. *answer generation* – the stage in which candidate answers are combined based on notions of similarity and overlap, and then scored according to overall correctness confidence. The final ordered answer set is presented to the user.

There are systems that allow feedback loops [42] among components when more information content such as documents, answers etc is needed. Planning [94] is also used as a tool to control the information flow between components and to guide the question answering process to better results. For example if the extraction stage in the question answering process cannot extract high confidence answers, a question answering planner might implement a recovery strategy that would require the retrieval stage to obtain additional documents, or the analysis stage to provide additional information (e.g. lower probability expected answer types) about the question or the expected answer.

There are several main dimensions to questions and answers. Questions can be classified into simple (factoid) and more complex, they can be open-domain or close domain, and their answers can come from the Web or from other corpora (e.g. local corpora). Depending on the specific languages they are tailored to, systems also cover a wide spectrum in terms of the resources and processing tools they are built upon, as well as their structure. For some languages, parsing and named entity extraction might be highly dependable, while for other languages they might be insufficiently accurate to be used as building blocks within question answering systems.

Questions whose answers are simple, concisely stated facts are called *factoid* questions (e.g. *Who killed Kennedy?, Where is London? How hot is the center of the sun?*) Non-factoid questions, which are sometimes ambiguously labeled “complex questions”, usually accept answers that are longer and more involved: definitional questions (e.g. *What is an atom? Who is Colin Powell?*), explanation requests and proofs (e.g. *Why is the Earth round?*), process questions (e.g. *How does blood coagulate? How do rainbows form?*). FAQ type questions are usually a mix of simple and complex questions such as the ones described above, and are usually answered by longer paragraphs.

1.1 Issues in Question Answering

Ever since Question Answering (QA) emerged as an active research field, the community has slowly diversified question types, increased question complexity, and refined evaluation metrics - as reflected by the TREC QA track [125]. Starting from successful pipeline architectures [87, 49, 21], QA systems have responded to changes in the nature of the QA task by incorporating knowledge resources [46, 52], handling additional types of questions, employing complex reasoning mechanisms [85, 93], tapping into external data sources such as the Web, encyclopedias, databases [31, 132], and merging multiple agents and strategies into meta-systems [18, 17].

Many successful systems have been built through many expert-hours dedicated to improve question parsing, question ontologies, question type dependent query structure and content, rules for answer extraction/selection, as well as answer clustering, composition, and scoring. Moreover, with the effort dedicated to improving monolingual system performance, system parameters are very well tuned. These aspects make training of components in many question answering systems very time-consuming and hard to train.

The QA community has acquired training questions and corresponding correct answers from past official question answering evaluations. One of the problems researchers in question answering face is the fact that the known correct answer sets are not complete: i.e. for many questions there exist other correct answers not part of the answer set. Moreover, answers can be reformulated in countless ways. Another issue is the extent of the answer. Consider the question “*What is the longest river in the US?*”. The extent of the answer “*Missouri*

River" is considered appropriate for the TREC evaluation, whereas the extent of the answer "*In the United States, the longest river is the Missouri River*", although perfectly reasonable for human consumption, is not considered appropriate by rigid evaluation guidelines.

Furthermore, answer correctness is often considered to be a binary decision. In reality, each answer may have a different degree of relevance to the question, may be partially correct, may provide relevant and useful information to the user even if it does not contain all the sought-after elements, or may have a different granularity (e.g. a *nematode* is a *worm*, but it is also an *invertebrate* and an *animal*). Answers also have a time component which can render them correct if the corresponding question is asked at one point in time and incorrect if the question is asked at another point in time. For example, the question "*Who is the president of the United States?*" might have a different answer every four years. In addition to time dependency, since question answering systems are often tuned to work with data from specific corpora (e.g. the Web or a particular local corpus), the tuned techniques work better on these specific corpora than on other document sources. This translates into a bias towards finding more answers from some sources (i.e. text collections) rather than others.

Due to specialized applications and standardized evaluation, many question answering systems are trained to perform well on questions from a particular language (i.e. English) and for particular domains. The questions provided by past TREC evaluations are considered to be *open-domain* questions. However most of them are acquired from web logs and reflect a main-stream pop culture interest, and are not more uniformly distributed across domains. Hence, in order to port them to other languages and domains, considerable effort is required. Furthermore, resources such as WordNet [82] and gazetteers have different coverage in different languages and may have a strong bias towards United States-centric knowledge. Processing tools such as parsers, part of speech taggers, and named entity taggers have different error rates for different languages. Because of these problems, it is very difficult to use the same perfected methods, tools, and expertise, and build question answering systems that are successful in new environments.

1.2 Statistical Elements in Question Answering

In recent years, learning components have started to permeate Question Answering [19, 106, 32]. Although the field is still dominated by knowledge-intensive approaches, components such as question classification, answer extraction, and answer verification are beginning to be addressed through statistical methods. At the same time, research efforts in data acquisition promise to deliver increasingly larger question-answer datasets [38, 33]. Moreover, question answering systems have been built for different languages [75, 76] and domains – other than news stories [137]. These trends suggest the need for principled, statistically based, easily re-trainable, language independent question answering systems that take full advantage of large amounts of training data.

Statistical components in question answering require more training data than rule-based and knowledge-based components, which rely more on generalizable expert knowledge. Training data for question answering consists of questions and correct answer pairs in the simplest form and also of known relevant documents, known relevant passages, high precision pattern sets for specific answer types. Because of the increasing need of training data, and the cost and effort involved in manually obtaining it, current efforts in automatic data acquisition for question answering are becoming more and more common. For example, a supervised algorithm acquired part-whole relations [38] to be used in answer extraction. The relations were based on 20,000 manually inspected sentences and on 53,944 manually annotated relations. The same research proposes a supervised algorithm [33] that uses part of speech patterns and a large corpus to extract semantic relations for *Who-is* type questions and builds an offline question-answer database. The database is then used for answer extraction within a more complex question answering system.

Training questions and answers provide the basis for statistical components in QA systems. The more similar the distribution of training questions is to the distribution of test questions, the better the systems perform. Currently, question and answer datasets are small and provide limited training data points for statistical components. In recent research [70] we have shown the viability of QA data acquisition from local corpora in an semi-supervised fashion. Such efforts promise to provide large and dense datasets required by instance based approaches.

Several statistical approaches have proven to be successful in answer extraction. The statistical agent presented in [18] uses maximum entropy and models answer correctness by introducing a hidden variable representing the expected answer type. Large corpora such as the Web can be mined for simple patterns [106] corresponding to individual question types. These patterns are then applied to test questions in order to extract answers. Other methods rely solely on answer redundancy [31]: high performance retrieval engines and large corpora contribute to the fact that the most redundant entity is very often the correct answer.

1.3 Question Answering in Specific Domains and Languages

Until recently, restricted domains were used in information extraction in order to construct templates for specific actions and entities fulfilling specific roles. However, with recent advances in question answering for the news domain, researchers have largely ignored issues pertaining to building QA systems for restricted domains. The 42^{nd} Annual Meeting of the Association for Computational Linguistics (ACL) has hosted a workshop on question answering in restricted domains, which took some preliminary steps in establishing basic research problems specific to domains other than news or pop-culture.

When applied to technical domains [88, 107], question answering faces various problems that are less prominent when building open-domain question answering systems. For example, in technical domains ambiguity in question formulation might be greater if users are less familiar with terminology and it is harder to generate focused queries. However, if queries are built successfully according to the user's need, there is the potential for less ambiguity due to the specificity of terms in technical domains, which have a lower average of meanings at the word level – i.e. less interpretability.

Medical text collections are becoming increasingly larger and the number medical knowledge resources is growing. Information retrieval and question answering [137] are starting to address information access problems particular to this field. Semantic classes of expected answer types are very different for medical domain questions than for open-domain questions with answers found in news corpora:. For example disease, medication, patient symptoms, and treatment outcome are more frequent in the medical domain. Recent research has shown that current technologies for factoid question answering are not adequate for clinical questions [92, 91]. Preliminary research in clinical question answering has approached the problem by exploiting domain specific semantic classes and the relationships among them. Semantic classes are further used to find potential answers and support vector machine classifiers are employed to label the outcome: positive versus negative.

Since much of the evaluation of open-domain questions has been done using local corpora consisting of news stories, an interesting study [36] analyzes different features between scientific text and journalistic text. They argue that indicators such as structure, past tense usage, voice and stylistic conventions affect the question answering process differently in these two domains.

Another domain to which people have started to adapt question answering systems is the genomics domain. Scientific documents in the genomics domain contain different terminology that may appear with its secondary meaning in open-domain resources. Differences in meaning, which are often quantified in terms of differences in WordNet synset ids, may result in different query content during document retrieval, and different rules and models for answer extraction. ExtrAns [107] is a QA system designed for terminology-rich domains which performs deep linguistic analysis and transforms documents into logical forms offline. Beyond the greater ambiguities in question formulations, additional problems consist of particularities of text collections: document type, manual or automatic annotations (if any), and stylistic and notational differences in technical terms.

Monolingual question answering is an active field of research not only in English, but in other languages as well. The Cross-Language Evaluation Forum (CLEF) is a forum in which cross language retrieval systems and question answering systems are tested for various European languages. The CLEF QA monolingual task started in 2003 with three languages and successfully progressed in 2004 to six languages: Dutch, French, German, Italian, Portuguese, and Spanish. The evaluation was performed using for each language 200 factoid questions which required exact answer strings and approximately 10% were definitional questions. Also in recent years the NII-NACSIS Test Collection for IR Systems project (NTCIR) has pioneered a series of cross-lingual and monolingual tasks [35] for the Chinese, Japanese, Korean, and English languages. These evaluations are becoming increasingly important since they are encouraging portable question answering systems – both monolingual and cross-lingual. Furthermore, the training data provided by these evaluations can be used to improve the performance of data-driven question answering systems with statistical components.

CHAPTER 2

IBQA Contributions

Thesis Hypothesis: *Question answering be done fully automatically, without a human in the loop during testing and training. Such an approach can rely only on statistical methods and use only (question, answer) pairs as the raw data. It is possible for such an approach allow to rigorous component-level evaluation and moreover, such an approach would achieve good performance, comparable to top systems in official evaluations.*

In this research we investigate the feasibility of an instance-based question answering approach in which answering strategies are derived directly from raw data – questions and correct answers. Can the performance of an instance-based QA system improve with more data? Are confidence scores produced through such an approach correlated with answer correctness? What is the necessary quantity and density of training data required in order to obtain meaningful answers to questions? What are the trade-offs among human expertise, resources, training time, and performance for such an approach? Can a resource non-intensive statistical approach constitute a good basis for a QA system, easily retrainable for different languages and domains?

These are some of the questions we attempt to answer in this research. In the process of presenting an instance-based statistical QA approach we examine some of the question answering issues raised above (section 1.1) and propose more flexible solutions: maintaining the probabilistic nature of answer types, learning query content from similar successful questions, constructing answer extractors from clusters of similar questions.

We present a principled, data-driven, **instance-based** (**IBQA**) approach to question answering that learns multiple answering strategies directly from clusters of similar training questions. The IBQA approach obtains good performance on factoid and definitional questions, comparable to the performance of top systems participating in official question answering evaluations. More specifically, the contributions of the instance-based QA approach consist of:

- **question clustering –** under IBQA, question analysis and classification are based on clusters of questions rather than based on answer/question type ontologies. Answering strategies are directly learned directly from these clusters.

- **multiple strategies –** individual strategies are learned from individual clusters of different granularity, scatter, and size. The relevance of a new cluster varies depending on the question we are trying to answer. Since a new question can belong to several clusters, multiple cluster-specific strategies are simultaneously employed, each contributing to the final set of answers.

- **resource non-intensive –** the core instance-based approach does not rely on resources such as: WordNet, parsers, taggers, ontology, hand-coded optimizations, and hand-coded patterns. However, the approach is resource-friendly, allowing external resources to be incorporated into an instance-based QA system.

- **fully statistical –** each stage in the question answering process is data-driven and a measure of the probability of success is directly incorporated in the overall answer score, rather than making hard local decisions.

- **data driven –** training question datasets dictate what question clusters are formed and how accurate the answering strategies are when they are learned from these clusters. The document corpora also directly influence what models are learned and what type of questions can be successfully answered or not.

- **learn query strategies –** from each cluster of training questions we automatically derive additional query content in order to focus and enhance queries, and consequently improve the likelihood of success of retrieval in the QA process.

- **question type independent –** since training questions guide the answering strategy learning process, the instance-based approach can be applied to more than factoid questions. Towards this end, we show experiments with definitional questions.

- **domain independent –** state of the art question answering systems employ domain specific elements: rules, query enhancements, and heuristics that are highly dependent on assumptions based on the content and format of questions and data available. The core instance-based approach does not rely on domain specific components and allows the training questions and the raw data to shape the answering strategies.

- **language independent –** the core instance-based question answering approach is language independent and can easily be re-trained for individual languages. Although the approach does not depend on language-specific resources or manual parameter optimization it allows the integration of language-dependent tools: part of speech tagging, parsing, and named entity tagging.

- **fast re-training –** the IBQA approach fully trainable and is not based on hand-written rules and hand-tuned parameters. This allows for fast re-training, which requires minimum human effort.

- **scalability –** depending on the clustering algorithms, the size and distribution of the training dataset, an instance-based QA system that fully explores all available strategies can be very slow. By selecting a small number of strategies according to confidence scores, we observe a limited overall performance degradation.

CHAPTER 3

An Instance-Based Approach to Question Answering

Traditionally, researchers have developed question answering systems by observing large sets of similar questions and constructing sequences of specific, carefully implemented steps designed to lead to correct answers. The initial approaches consisted of observing the most frequent types of questions and focusing on devising a pipeline of models to analyze the questions, retrieve good documents, extract answers, ranking them, and presenting them to the user. This process is typically tedious and involves expertise in crafting and implementing these models (e.g. rule-based), utilizing NLP resources, and optimizing every stage *for every question type* that occurs frequently[1]. Several systems have started to employ statistical models for each stage in this pipeline and have also started to improve the feedback, interface, and control among these modules. However, there is still a high degree of complexity required in tuning these systems, tailoring them to the TREC/CLEF domains, English language, and making sure that multiple strategies (increasingly more common) are selected and ordered as close to optimal as possible.

[1] most systems are still optimized for TREC and CLEF question types

To fill this void and to provide a more robust, adaptive baseline, we require a data-driven approach, capable of taking advantage of this mechanism. Such a data-driven approach should attempt to automate the question answering process as a whole, by allowing different datasets of training questions to guide the learning process in terms of retrieval, extraction, and answer generation – i.e. the critical stages in a QA pipeline.

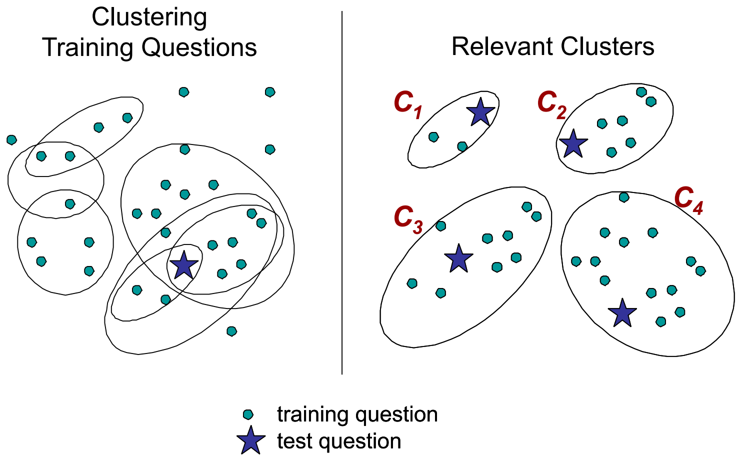

Figure 3.1: Training questions are clustered according to some criterion and shown in a bi-dimensional projection of the multi-dimensional feature space. Test questions are also represented in this space. Relevant clusters of similar questions are identified and their corresponding models are applied in order to find correct answers.

We propose an *instance-based, data-driven* (IBQA) approach to question answering. We adopt the view that strategies required in answering new questions can be directly learned [69] from similar training examples: question-answer pairs. Instead of classifying questions according to limited, predefined ontologies, we allow training data to shape the models and ensure they are capable of answering new similar questions. Towards this end, we propose **clustering training questions** in order to learn more focused models. Answering strategies consisting of answer models, query content models, and extraction models are learned directly from each individual cluster of training questions. To answer new questions, multiple clusters are identified as relevant and their corresponding answering strategies are activated. In order to maintain a general, accessible approach, we designed our framework to be compatible with existing components of question answering systems – e.g. QA ontologies, query types and query processing, answer extractors, and answer merging methods.

In this chapter we describe the general IBQA framework and provide a high level description of the relevant stages/components. This framework allows different component implementation using various methods and algorithms. Here, we focus on defining the stage-specific tasks and providing an overview of the IBQA framework. In future chapters we discuss specific component and end-to-end implementation.

Consider a multi-dimensional space, determined by features (e.g. lexical, syntactic, semantic, surface form) that can be extracted from questions. In this feature space we project the training questions, representing each instance as a data point (vector of feature values). In this space, we cluster the questions (Figure 3.1) with the purpose of obtaining sets of training data that are more homogeneous and from which we can learn useful answering strategies. If we use all the training data and attempt to learn one answering strategy, the diversity of questions and possible approaches is overwhelming. Through clustering, the goal is to reduce this noise and provide datasets of similar questions that may be processed in a QA system using a cluster-specific, dedicated answering strategy.

In this multi-dimensional space, features can range from lexical n-grams to parse tree elements, depending on the available processing tools and also on implementation complexity. Test questions are also represented in this feature space and cluster relevance is computed as the distance to individual cluster centroids. Although in this work we show several methods for implementing feature extraction and clustering, the instance-based QA framework is independent on the type of clustering and on the dimensions chosen: e.g. semantic representation, syntactic representation, surface form representation, user profile, question statistics, corpus statistics, topic, question source etc.

An alternate way to view the IBQA approach is as a nearest neighbor classification using clusters of training questions as the test question's neighborhood. Clustering allows us to overcome the sparsity of the data and to acknowledge that different clusters of similar training questions capture different aspects of the test question. In other words many questions can be similar, but they can be similar according to different dimensions. Simultaneously exploiting different types of similarity is key to generating multiple strategies and using them to attempt to answer the same question in many different ways.

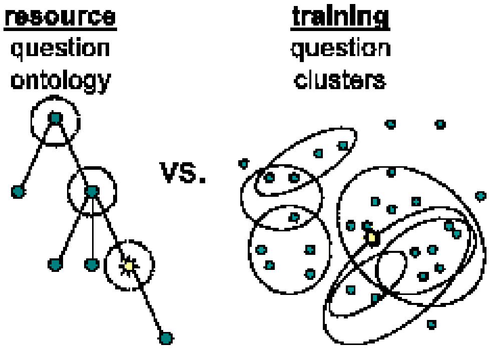

Figure 3.2: Classification according to a question ontology versus classification according to a set of clusters in the training data. For a new question, multiple question types (ontology nodes) correspond to multiple clusters. The use of answering strategies corresponding to different clusters is equivalent to the use of answering strategies corresponding to different ontology nodes.

From a more traditional perspective, clusters can be thought of as question types (Figure 3.2). These question types are derived dynamically based on similarity. They can also have different granularity and they are not required to be disjoint – as is very often the case in question type ontologies. This view is similar to a question ontology except cluster overlap is allowed. Moreover, a question is assigned multiple types (i.e. belongs to multiple clusters) and these types can be of varying granularity.

Under the instance-based QA approach, clusters may differ in granularity, number of data points, and scatter. When a test question is similar to training questions according to several clusters, multiple answering strategies are employed (Figure 3.3), each producing a cluster-specific answer set. These answer sets are then merged and an overall answer set is generated. We train an overall answer generation model that combines evidence from individual sets and clusters the answers based on specificity, type, frequency, and confidence.

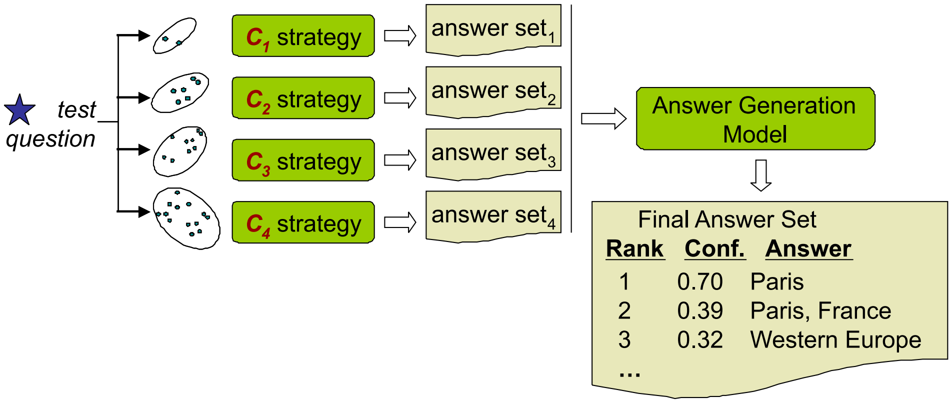

Figure 3.3: Multiple answering strategies are activated concomitantly depending on test question similarity to training questions. An overall *Answer Generation Model* is learned from the training data in order to merge individual answer sets produced by cluster-specific strategies, compute the final confidence scores, and generate the final answer set. Note that these strategies are based on heterogeneous clusters (different sizes, granularities, cohesiveness etc).

3.1 Answering Strategies

Most question answering systems are implemented as a pipeline where different stages successively process data. However, for each stage in the QA pipeline there is a variety of methods that can be employed. Each method typically has different parameters, needs different resources, and may produce answers with different confidences. These confidence scores may not be comparable across methods. We will refer to a complete combination of components at each stage in the pipeline as an *answering strategy*. In most of today's QA systems, an **answering strategy** consists of the following components:

1. **question analysis** – produces an expected answer type, extracts question keywords, and analyzes the question. Part of speech tagging, parsing, semantic analysis and additional processing are sometimes used in question analysis.

2. **retrieval** – specifies what query types and what query content yield high expected performance. Very often QA systems manually pre-specify the query type and additional content according to the question and answer types identified earlier in the strategy.

3. **answer extraction** – specifies how answers are identified from relevant documents. Answer extraction methods range from rule and pattern-based extractors to hidden markov models (HMM), maximum entropy, and support vector machine-based extractors.

When applied to a new question, an answering strategy processes the question text, retrieves documents and extracts a set of possible answers. In the case when multiple strategies are simultaneously applied to a new question, an **answer merging** component is employed to combine answers and confidences into a final answer set:

4. **answer merging** – combines the answers obtained through multiple answering strategies (stages 1-3). Multiple occurrences of the same answer with different confidence scores are combined. Note that the answer merging component is not actually part of any specific answering strategy.

Table 3.1 shows two simplistic strategies for the question "When did Mozart die?". In realistic scenarios the question analysis component produces more information than just an expected answer type, several queries are generated according to pre-specified types, and various processing is performed before answer extraction.

As the first stage in answering strategies, most question answering systems employ question ontologies. These ontologies combine expected answer types (date, location etc) and question types (*birthday(X)*, *nickname(X)*, *construction_date(X)* etc). Consider again the question "When did Mozart die?". Depending on the desired answer type granularity, this question can be classified as a *temporal* question, a *temporal::year* question, or more specifically as a *temporal::year::death_year* question. Each classification may lead to an entirely different answering strategy. Existing systems consider answer types ranging from simple answer type sets and QA specific ontologies to semantic networks such as WordNet, which provide better coverage and more specificity. However, these ontologies are very restrictive and only take into account the answer type, disregarding question structure, or domain knowledge.

Question: *When did Mozart die?*

QA Stage	**Strategy S_A**	**Strategy S_B**
1) analysis (answer type)	temporal expression	temporal::date::year
2) retrieval (queries)	• *when mozart die*	• *mozart die biography* • *mozart died death*
3) extraction (model)	• rule-based • SVM extractor	• HMM

Table 3.1: Answering strategies S_A and S_B use different answer types, different queries, and different extraction methods. These strategies may be generated by two different QA systems or by a multi-strategy question answering system.
The retrieval component for S_B is based on a more complex model the model used by strategy S_A. The S_B strategy expands on the question keywords, while the S_A strategy does not. The extraction methods for S_A is a combination of a rule-based extractor and an SVM extractor, while the extraction method for S_B is HMM-based.

The instance-based QA clustering approach [69] is in some respects similar to ontology-based approaches. Under IBQA training questions are clustered according to different similarity criteria such as shared number of n-grams (contiguous sequences of words), semantic similarity, and same answer type. Compared to fixed ontologies, this approach is adaptive to training data, is language and domain independent, and allows overlapping types (clusters) that do not have a hierarchical relationship. Figure 3.2 shows the relationship between ontology and clustering-based approaches for QA as they are used in question analysis (stage 1) of a QA process. If clustering is performed at different granularities, each cluster corresponds to an ontology node. Thus, individual answering strategies are built for different clusters, rather than different ontology nodes.

The clustering approach allows each component in an answering strategy to be learned only from i) training questions and ii) their known correct answers. Therefore strategies are learned for individual clusters, using corresponding questions as training data. The retrieval component learns which queries and query types have high performance when run on in-cluster training questions. The answer extraction component is trained on correct answers for all in-cluster questions. Finally, the answer merging component considers cluster statistics, retrieval performance, extraction performance, and merges answer sets produced by answering strategies.

If there is sufficient data for learning (i.e. sufficient number of questions), the more clusters of training questions a QA system generates, the more answering strategies will be applied to new questions. However, while QA performance may increase with additional answering strategies, so will the noise (e.g. from irrelevant clusters) and the time it takes to actually run these strategies. Our goal is to allow the existence of multiple cluster-based strategies, but only select a set of clusters associated to the strategies most likely to lead to high performance. For document retrieval, high performance translates into high recall of relevant documents. For answer extraction, high performance corresponds to a large number of correct answers being extracted.

Queries learned by different strategies often lead to some of the same relevant documents – e.g. the queries "*the first aria composed Mozart*" vs. "*aria Mozart*" may lead to an overlap in their retrieved document sets. If a strategy already leads to the retrieval of a document d_i, subsequent strategies will not benefit if they retrieve d_i again. Therefore, each strategy selection depends on the n-1 previously selected strategies.

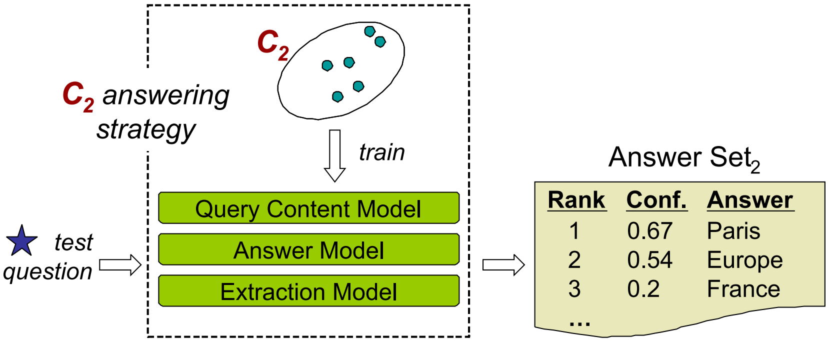

Figure 3.4: From each cluster of training questions (e.g. C_2) an answering strategy is derived in order to help answer similar new questions. Specific models are learned directly from training data in individual clusters: a *Query Content Model*, an *Answer Model*, and an *Extraction Model*. These models are jointly employed to answer new test questions and produce corresponding answers.

Figure 3.4 shows a test question and several answering strategies constructed directly from clusters that include it. As specified above, a strategy learning from training questions involves several steps: learning the distribution of the expected answer type, learning the structure and content of queries, as well as learning how to extract the answer. Although present in most question answering systems, these steps are often static, manually defined, or based on limited resources. In the instance-based approach, an answering strategy consisting of cluster-specific models can be fully learned. Under the IBQA framework, we package the main components of each strategy as well as the answer merging stage into four generic models whose implementation depends on individual QA systems:

1. the **Answer Model** $\mathcal{A}_i$ learns the cluster-specific distribution of answer types.

2. the **Query Content Model** $\mathcal{U}_i$ is trained to enhance the keyword-based queries with cluster-specific content conducive to better document retrieval. This model is orthogonal to query expansion.

3. the **Extraction Model** $\mathcal{E}_i$ is dynamically built for answer candidate extraction, by classifying snippets of text whether they contain a correct answer or not.

4. the **Answer Merging** (also refered to as "Answer Generation") $\mathcal{M}$ learns from training question/answer pairs to combine answers and confidence scores from different strategies.

Under IBQA, each of these models can be implemented using many different methods. By propagating and processing the training questions and their known correct answers through each of these models, we construct an answering strategy. Thus, specific models are built on the assumption that the original training questions are similar and therefore can be answered using a single shared strategy, which we are attempting to learn. Since this assumption may not be true given a particular cluster, we construct different strategies for different clusters to increase the likelihood that one of the strategies can produce correct answers with high confidence. The goal is to obtain high confidence strategies from clusters that do match our assumption and low confidence strategies from clusters that do not.

3.2 Scalability: Multiple Strategies & Strategy Selection

In the past few years, an increasing number of question answering systems have started employing *multi-strategy approaches* that attempt to complement one another when searching for answers to questions. These approaches sometimes include multiple question classifications, several retrieval approaches, multiple answer extractors, and different data sources. Question answering performance is often presented within the context of official evaluations where systems are processing batches of questions with no time constraints. However, in real-life scenarios, only a limited number of these strategies (component combinations, parameter settings, etc) can be fully explored. In these cases, the trade-off between performance and problem complexity (and indirectly response time) require careful selection of answering strategies such that performance is optimized according to realistic constraints.

In this dissertation we closely examine individual stages in an IBQA answering strategy. For each of these stages, we also investigate an *answering strategy selection* [71] approach that directly addresses the performance-complexity trade-off. We apply this selection strategy to our statistical, instance-based question answering system to explore the scalability of our IBQA framework. We investigate the benefits of a principled strategy selection method when applied to the main components of a QA system: document retrieval, answer extraction, and answer merging (i.e. overall QA performance). Experiments show that by carefully selecting less than 10% of the available answering strategies, no significant performance degradation is observed. Moreover, we integrate a cluster-based confidence scoring method with an answer merging component and observe significant question answering performance improvements.

When considering the overall answering strategy performance, one has to examine the pipeline relationship between retrieval and extraction [23] and test the correlation between improved document retrieval performance, improved extraction performance, and overall QA accuracy. In our instance-based approach we incorporate retrieval and extraction confidence scores into our answer generation model.

Several practical approaches have been developed to deal with the complexity of the question answering process. The SMU system [42] and later the LCC system [86] incorporate feedback loops between components of their question answering system. The CMU system treats the QA process as planning problem, formalizing the notion of feedback. Several other QA systems using statistical components [18, 93, 69] introduced multiple answering strategies that can be used simultaneously and their results can be combined. Furthermore, when answering complex questions, [40] argue for a multi-strategy approach for question processing, extraction, and selection.

The strategy selection problem is closely related to active learning, which explores the trade-off between performance and cost. While active learning algorithms suggest data for labeling by minimizing the expected error [109], in the problem of strategy selection, the goal is to reduce QA complexity by limiting the number of answering strategies while not increasing the error of the QA process.

IBQA Components

In the following chapters we describe individual components of our instance-based question answering approach in detail and present experiments and results at component level as well as at system level.

In chapter 4 we introduce evaluation metrics and methodology for individual QA components. Chapter 5 reviews several clustering methods and discusses their applicability and usefulness to question clustering. Chapter 6 discusses different methods for modeling the expected answer type and perform ontology-based classification or cluster-based classification.

Once an expected answer type distribution is identified, we investigate retrieval methods for IBQA (Chapter 7) as well as experiment with different query expansion techniques. We show their cumulative benefit and we also introduce an additional cluster-based expansion method. When query strategies are learned, we retrieve documents that are more likely to be relevant. Chapter 8 discusses different answer extraction techniques and presents experiments using three methods. The third answer strategy model, the answer merging/generation is presented in chapter 9.

CHAPTER 4

Evaluation Methodology

One of the strengths of the instance-based question answering approach is the ability to test and analyze the performance of individual components and modules under different conditions. Rather than only presenting overall system performance with certain parameter settings, we investigate in-depth component-level performance. Every part of an answering strategy can be evaluated individually using different metrics and according to different criteria. In-depth experiments can offer a better idea of model robustness, can uncover bottleneck components, and may provide a better understanding of the problem complexity, such that better design and implementation choices can be made.

In the remainder of the chapter we will describe the evaluation methodology for individual IBQA components and also for the end-to-end instance based system. Together with each component description, we also present focused local experiments and local component performance. These experiments also uncover how errors propagate in successive modules in the QA pipeline and explain the overall end-to-end performance (chapter 10). Before we explore in detail the instance-based QA framework, in order to understand how each component is evaluated, the following sections provide the necessary background into evaluation methodology, evaluation metrics, and component-level (answer modeling, document retrieval, answer extraction, and answer merging) criteria of success.

4.1 Metrics Used in Question Answering

In recent years, the TExt REtrieval Conference (TREC) environment [125] has been a standard evaluation forum for measuring the performance of question answering systems. This annual evaluation has focused the discussion of how to measure the success of a QA system. In general, the following are metrics employed in factoid question answering evaluations:

- **Percent Correct –** (same as Accuracy) for each question, only the first, highest scoring answer provided by a QA system is considered. Percent correct refers to the precision of a QA system over all N questions. This is a special case of a system making a binary decision whether there is a correct answer in the top K candidate answers proposed by a system.

  $$\text{Percent Correct} = \frac{100}{N} \cdot \sum_{i=1}^{N} c(A_{i1}) \tag{4.1}$$

  where $c(A_{i1})$ is an indicator function which takes the value 1 if the first answer to question Q_i is correct and the value 0 if the A_{1i} is incorrect.

- **Percent Correct in TopK –** a question is answered successfully if there is at least one correct answer in the top K answers provided by a question answering system. The overall system score is the average precision of all questions.

  $$\text{Percent Correct} = \frac{100}{N} \cdot \sum_{i=1}^{N} (c(A_{i1}) \vee c(A_{i2}) \vee \cdots \vee c(A_{iK})) \tag{4.2}$$

- **Mean Reciprocal Rank (MRR) –** probably the most widely used metric, it computes the average reciprocal rank (RR) of the first correct answer for each question. Usually only the top 5 answers are considered when scoring each question – if the first correct answer is not in the top five answers, the RR for the question is 0

  $$MRR = \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{1}{\text{first correct answer rank}_i} \tag{4.3}$$

  MRR captures correct answers that are in the top five. While the weighting scheme has been questioned before (e.g. the second answer is only assigned a 0.5 score), it is still widely used and very useful in evaluating question answering performance.

- **Confidence Weighted Score (CWS) –** measure combining the percent of correct answers and the confidence of the system in its scoring method. Questions are ordered according to confidence in their corresponding answers.

  $$CWS = \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{\sum_{j=0}^{i} c(A_{j1})}{i} \tag{4.4}$$

  CWS measures a combination between the average objective performance of the system and the quality of the confidence of the system in its answers. Two QA systems that answer a set of questions with exactly the same answers, yet order the questions differently may obtain different confidence weighted scores.

Within the TREC evaluation mean reciprocal rank was one of the earlier scoring metrics and is still used to report QA performance. It has the advantage of incorporating information about the rank of the first correct answer from all questions into a single performance score. The main drawback is the function used to estimate the utility between two ranks – i.e. should a rank two answer be assigned half the score or of a rank one answer? MRR was used for several years even though some researchers proposed a linear function. This weighting scheme also encouraged systems to focus on obtaining correct answers with rank one. Confidence-weighted score and percent correct have been more recently frequently used to characterize the performance of QA systems and disregard possible correct answers unless they have rank 1 in the answer set. The CWS metric also encourages systems to generate scores that correlate well with the actual system performance.

While there are types of complex questions (e.g. questions that require inference or combining answer from multiple documents), that may have correct answers in the form of factoids, several types of non-factoid questions questions require more involved answers. For example definitional questions such as "*What is thalassemia?*" or "*Who is Desmond Tutu*" may require answers that are longer than simple factoids. In our example, Desmond Tutu is a Nobel Peace Laureate, an archbishop, a South African, a master of theology, a professor (dean), but also an author and a husband. These bits of information can be incorporated into a huge number of correct answers formulated differently. Question types such as FAQ, how-to (e.g. "*How is the process of fusion applied in practice?*"), and why-questions (e.g. "*Why did North Korea decide to pursue a nuclear program?*" are very difficult to answer and certainly require more involved reasoning, processing, and verification.

The following metrics can be used to evaluate the performance of definitional (and other complex) questions. They assume that several text segments (called nuggets) are identified as being *relevant* and can be used as features to precision/recall-based metrics. Relevance of textual nuggets is measured against a set of textual 'keys' (nuggets themselves) known to be correct. In an earlier example we considered the question: "*Who is Desmond Tutu?*". For this example, the the answer nuggets "*archbishop*", "*Nobel Peace Prize*" , and "*author*" would be considered as relevant since they reflect defining aspects of the question's target, Desmond Tutu. These nuggets are part of a larger set of known relevant nuggets. Precision and recall are measured over these set of *nuggets* (text snippets), assumed to collectively capture the correct answer set for a question. The commonly used metrics are based on the fraction of correct nuggets captured by a system's answers as well as a system's answer precision:

- **Nugget Recall –** simple recall is a very good measure of what fraction of nuggets are covered by the top single answer. If the answer size is limited (e.g. 50 words), nugget recall may be a reasonable measure.

- **Nugget F-measure** – used in evaluating definitional questions, nugget f-measure is a metric that combines precision and recall using a weighted harmonic mean [120]

$$\text{F-measure} = \frac{(\beta^2 + 1)RP}{\beta^2 P + R} \tag{4.5}$$

  where β is a parameter signifying the relative importance of precision P versus recall R. In past TREC evaluations a β parameter that emphasises recall over precision has been used (i.e. 3-5 times more), with the goal of capturing as many of the relevant nuggets as possible, at the cost of lower precision.

- **Rouge** – automatic evaluation metric based on n-grams and defined in the context of text summarization that compares a new answer with existing reference answers. However, in order for Rouge to be useful, several re-formulations of reference answers need to be created [133]. Also, since this method is n-gram based, a drawback is that reference answers need to be longer to capture higher-level n-grams. Originally, Rouge measured the overlap overlap between automatic summaries (candidates) and manual summaries (reference), at n-gram, or word-level. For question answering, the known correct answer nugget set is the reference and the system produced answers are the candidates. Since it is not a standard QA measure, we will not use Rouge in this work – for more details about Rouge in question answering, please refer to [133].

- **Keyword Overlap** – several rough automatic metrics based on weighted keyword overlap matching have surfaced, they attempt to adapt ideas from machine translation (i.e. Bleu [96]) and summarization (i.e. Rouge [65]) to question answering. Most notably **Pourpre** [66] is based on the assumption that the fluency criterion in machine translation does not apply to question answering, hence the goal has to be measuring the adequacy - summing unigram co-occurrences between nuggets and system response. **Qaviar** [13] computes the recall against the stemmed content words in human generated answer key. **Nuggeteer** [80] is a similar metric that is perhaps better suited for error analysis and also allows expandability of answer keys, and easier interpretability of scores.

- **N-gram Co-occurrence** – for FAQ type questions [110] the goal is to find segments of text (e.g. paragraphs) that overlap as much as possible with known FAQ answers (truth). The more overlap they have, the better the retrieved answer segment is. However, for definitional questions, measuring relevant nugget overlap is a better measure than comparing answers to a long narrative reference (e.g. Desmond Tutu's biography).

Several approaches to defining nuggets have been proposed for definitional question answering. However, automatic nugget-based metrics are still problematic. The first issue has to do with nugget relevance – i.e. not all nuggets are equally relevant to the question. For example, in most contexts the fact that Desmond Tutu was a *husband* should not be as important as the fact that he was an *archbishop*. This can be seen as an information gain measure, meaning that we aim for a large decrease in entropy when a correct answer nugget is known. This observation led to the division of nuggets into *vital* and *okay*. Although this division is artificial and does not fully quantify the relevance of individual nuggets, it attempts to differentiate between critical information and useful information, in the context of TREC.

Another problem with nugget-based evaluation reflects a more general problem with using answer keys to evaluate system performance. The coverage of answer nuggets is very low, especially when taking into account variations in surface-form. This means that given an answer nugget of a certain form (e.g. "*Nobel Peace Laureate*") and a system answer of a different form but near identical meaning ("*Nobel Peace Prize Winner*" or "*Nobel Peace Prize*"), a rough fully automatic method would conclude they are different. Although there are methods that also take into account nugget-answer overlap in terms of constituent n-grams, semantic overlap, and morphological variation, it is very difficult to overcome this problem. Currently NIST uses human assessors to evaluate definitional questions for official TREC runs. Clearly, further refinement of nugget definition and automatic evaluation methods are required.

4.2 Component-Based Evaluation

Very often, question answering systems present a single number performance score over a certain dataset. This type of performance, however useful is not very revealing of what algorithms, implementation details, data sources, and resources are responsible for that result. Typical question answering systems are intricate, complex software engineering endeavors that bring together several fields such as natural language processing and machine learning. With different QA systems implementing some of the same algorithms and using some of the same methods for question analysis, document retrieval, answer extraction, and answer merging, it becomes necessary to identify the particular component or resource that is responsible for significant performance improvements. Some research systems have already started to build QA platforms that are open to error analysis and component-level investigation. The Javelin CMU system [93] was the first to implement a robust, modular architecture that accommodates all the traditional QA pipeline components, and also planning, reasoning, verification, and user interaction.

Our instance-based approach implements a modular architecture as well, allowing component-level evaluation. Because of its data-driven nature, it also accommodates a greater level of component-level training for different datasets. We investigate each component in our IBQA pipeline and perform component-level experiments using a focused dataset consisting of temporal questions from TREC evaluations. This offers a compact dataset feasible for in depth local evaluations. We experiment with different methods, parameters, and resources and evaluate performance for question clustering, document retrieval, answer extraction, and answer merging.

4.2.1 Answer Modeling Component

In the context of question analysis, answer modeling plays the critical role of identifying the characteristics of the expected answer. The basic task is to analyze the question and provide an expected semantic answer type. In most systems, document retrieval, answer extraction, and answer merging heavily rely on the expected answer type to focus their methods towards finding a correct answer of that particular type.

Under IBQA we argue for the need of answer type distributions and we perform local experiments to test the hypotheses that i) answer type distributions help, ii) even the simplest answer type distribution generating method improves with more data (or alternately, performance does not decrease by considering more similar questions), and iii) similarity weighting improves the answer type distribution method. We evaluate the answer modeling component using several measures of divergence (cosine similarity, KL distance, JS distance, and χ^2 between the generated expected answer type distribution and the true answer type distribution. We also argue that cluster-based distributions have a better chance of taking advantage of question similarity. However, the same method can be applied to question answering ontologies or answer type sets by computing the probability of an answer type occurring given a test question and a specific cluster of similar questions.

4.2.2 Document Retrieval Component

Most question answering systems directly or indirectly employ a search engine for the retrieval of relevant documents. Sophisticated retrieval components in QA consist of query building mechanisms that take question keywords and expand them through morphological and semantic methods. The queries are then structured and expanded with question type-related content and then run through a retrieval engine on a local corpus or on the web. Measuring retrieval performance in question answering goes beyond rank and precision computation. We define a document to be relevant if it contains a correct answer in a correct context. For practical considerations we relax the correctness definition to documents that contain a correct answer, regardless of context. In the case of retrieval under the instance-based framework, we are interested in obtaining relevant documents with high density. Relevance ranking is only important for experiments with constraints on the number of documents, thus ensuring that a relevant document is always in the top documents retrieved.

We evaluate the impact of several different query expansion methods on our IBQA system, both individually and additively. In particular, we are interested in evaluating the improvement in relevant document density of query expansion methods over simple keyword based queries, in the context of our instance-based approach. We compare synonym expansion, inflectional form expansion, and a cluster-based expansion method that we introduce in section 7.3. To take advantage of all of our limited QA data, the experiments using the cluster-based query expansion method were performed using leave-one-out cross validation.

Improved retrieval in question answering is critical so that further modules in the QA pipeline, especially answer extraction, have sufficient (redundant) text segments that contain correct answers appearing in various contexts. Therefore the more relevant documents are retrieved by the IR component, the higher the answer recall will be – i.e. the more likely it is for the correct answer to be extracted and supported by different contexts.

Since the instance-based approach is data-driven, we would like to be able to *automatically* add relevant content to queries that can help improve retrieval performance. To that end, we also evaluate several feature selection methods individually and cumulatively. Feature selection identifies content that has the highest potential for improving a query. A desirable property of feature selection specifies that the score associated with each content feature should be proportional to the actual retrieval performance – i.e. precision of the improved query. For example given the query "*mozart die*", it can be argued that the content feature "*biography*" should have a higher score (likelihood of improving retrieval) than the content feature "*chocolate*". Another desirable but not necessary property is correlation between rank and performance. For our IBQA implementation, we investigate the performance of various methods as a function of the number of documents retrieved. While this is also a measure of the intrinsic relevant corpus density and of the specific question set used, it also ensures the generalizability of our query expansion method.

4.2.3 Answer Extraction Component

The central component in a question answering system is answer extraction. The goal of the extraction stage is to identify potential answers in running text and score them according to how likely they are to be correct. The running text consists of documents or passages that have been retrieved by the previous stage in the pipeline. The assumption is that at least part of the documents given to the extraction component are relevant – i.e. contain a correct answer.

We experiment with three different extractors: proximity extractor, pattern-based extractor with automatically extracted patterns, and also with a support vector machine-based extraction method. These extractors are trained for TREC factoid questions. We evaluate the extractors using the mean reciprocal rank and correct in top k metrics. While both metrics offer an aggregate numeric score based on the several top answers, the TopK metric is more relevant for the extraction task.

Broad context complexity coverage is a property of the raw documents or passages presented to the answer extraction that is difficult to quantify and measure. If different answer extractors are exposed to correct answers in various contexts, they increase the likelihood of extracting at least one such answer. Towards this end, the retrieval stage employs various query expansion methods to obtain correct answers in different contexts. Simultaneously using multiple extraction methods, each with its own bias, also increases the chance of identifying correct answers in different contexts.

For each extraction method employed, we evaluate answer extractor performance as it varies with cluster size and cluster specificity. Lower cardinality clusters often do not have sufficient data to support learning of high precision models, but at the same time clusters that cover a large fraction of the training data may be too broad. Questions under such clusters tend to to have little in common with each other and therefore we cannot learn a strong cluster-specific strategy from them. Certainly, this is not a guarantee that clusters that have sufficient number of questions and have a moderate degree of similarity will generate strong, generalizable strategies.

These extraction experiments are performed under ideal conditions, where the answer type are already known. The noise is reduced by filtering sentences using potential answers of the appropriate answer type. In a typical QA system not all segments of text that are of the appropriate answer type will be selected and more noise will be introduced with potential answers that are not of the correct type, but which are considered by extractors. However, it is necessary to create these conditions so that errors from the question analysis and document retrieval do not propagate, in order to test answer extraction performance. We evaluate the performance of individual answer extractors under different conditions, independent of the rest of the QA system. In chapter 10 we integrate the answer extractor and evaluate the ensemble of all the components of our instance-based system.

4.2.4 Answer Merging

The simplest question answering system structure does not require an answer generation component. Answers are presented to the user as they are extracted from text. However, through answer clustering and merging, QA systems may benefit from answer redundancy, answer granularity, and quality of answer extraction in order to formulate complete answers, score them appropriately, and achieve a high correlation between answer correctness and answer confidence.

The instance-based approach, similar to most question answering systems, incorporates an answer merging component that attempts to combine multiple instances of the same answer and also combine their confidence scores. Although individual answer extraction scores are very relevant, multiple instance support often boosts the scores of correct answers, effectively modifying the ranking produced by answer extraction.

A straight-forward evaluation for the answer generation component is measuring the performance of the system with and without (baseline) using answer generation. The higher the performance gap, the more useful the answer generation component is to the question answering process. An intermediate step is to measure the performance after scores are normalized but before answers are clustered. This provides a better insight into system component behavior.

4.2.5 End-to-End Evaluation

On of the main applications of this work is to provide a flexible, trainable QA system for factoid question datasets. Current system building in question answering generally requires many expertise, annotation, data, and considerable implementation and parameter tuning. Providing an approach that is comparatively simpler to implement and that can automatically adapt to different datasets could be viewed as starting with a much higher baseline and with a more principled system with comparatively less effort. Moreover, strong factoid QA systems are the basis and the building blocks for constructing question answering systems capable of answering complex questions: i.e. list questions, questions that require reasoning, scenario-based questions etc. A strong and robust factoid question answering system would allow researchers to focus more on harder question types and less on parameter tuning, rule-based components, local component engineering.

We evaluate our IBQA system on a set of several thousand open domain[1] **factoid questions** from TREC evaluations (TREC-8 through TREC-13). Every year, the National Institute of Standards and Technology (NIST) has created a dataset of approximately 500 questions taken from real web logs of actual human-posed questions. Since the instance-based approach is data-driven, it requires some degree of redundancy to be able to train cluster-specific models. We use web documents as the underlying data source since they provide a higher degree of redundancy and variability in answer contexts compared to local corpora. Once built, these models can be used to answer new questions, whose supporting documents are drawn either from the web or from local corpora.

For each of the TREC factoid questions, a set of answer patterns in the form of regular expressions is available (thanks to Kenneth Litkowski). These regular expressions were generated through answers extracted from the AQUAINT corpus, and although they do not fully correlate with current web answers they still constitute an appropriate platform for a automatic evaluation of factoid questions. The AQUAINT corpus consists of newswire text data in English, acquired from three sources: the Xinhua News Service (People's Republic of China), the New York Times News Service, and the Associated Press Worldstream News Ser-

[1] most *open domain* QA work has been done on collections of news stories.

vice. It was prepared by the Linguistic Data Consortium (LDC) for the AQUAINT Project, and was used in official NIST's question answering yearly benchmark evaluations. An example of a problematic evaluation using these regular expressions is the question; "*Who is the current heavy weight champion?*". The AQUAINT corpus covers an earlier period (1999-2000) and answers extracted from it may not be accurate any longer. Therefore, matching a pattern created using the AQUAINT corpus on an answer extracted from web-documents will not necessarily provide us with an accurate evaluation. However, for most questions, these patterns work very well regardless of the corpus the answers come from.

We use the MRR and Top5 scoring metrics for the overall system performance and we experiment with different training set sizes and consider the cluster size effect over IBQA performance. We investigate whether performance does increase when more training data is provided, analyze the type of training data required, and discuss conditions under which saturation may occur.

Another issue closely related to system performance is QA **data acquisition**. Statistical systems rely heavily on available training data. In particular, the IBQA system requires questions and corresponding correct answers as the basis for question clustering and learning cluster-specific answering strategies. We evaluate our QA data acquisition experiments indirectly through the actual task of question answering. Since this can be viewed as simply a QA system performance evaluation, we use MRR and Top5 as the standard metrics. The goal of the evaluation is to show that by using our semi-supervised method for acquiring additional QA data similar to the existing training data, QA performance increases. However, data acquisition for question answering is in itself a difficult and current research problem. These experiments show how we can start exploring available data to enhance training data for our instance-based approach and for other statistical systems or QA components.

Definitional Questions

A goal of our IBQA approach is to have sufficient flexibility to apply it to a different question dataset that includes different question types. We move from factoid questions and apply the instance-based system to the set of all TREC **definitional questions**. We evaluate our end-to-end QA experiments using recall and the official NIST nugget-based scoring function. We also examine the differences between object definitions (e.g. "*What is thalassemia?*" and person profiles (e.g. "*Who is Colin Powell?*" in the IBQA context as they constitute the two major components of definitional questions.

Nugget-based evaluation of definition questions was developed initially by NIST [125, 126, 127] and it relies on lists of text fragment called *nuggets*, put together by official assessors. Initially, the answers from all of the QA systems that participated in the evaluation were presented to the assessor, together with searches done during the pre-evaluation stage when questions were put selected. Based on these resources, assessors compose a list of nuggets of relevant information that should be included textual in answers. The goal is for the nuggets to be objective components – as judged by assessors – of *correct* answers to definitional questions. Since nuggets are not always text segments that can be automatically matched in the text (e.g. the nugget is *Nobel Prize winner* while the answer includes *Nobel laureate*), the assessors have to ensure that during the evaluation phase, they can make a binary decision based on whether the nugget does or does not appear in the answer. Furthermore, the nugget is marked vital or non-vital, depending on the type of information it covers (desired vs. required). During evaluation, the assessors identify nuggets in each system's answer set. Using these nuggets, we employ the same method to evaluate our system's performance on definitional questions. Nugget recall R_{def} is considered to be the ratio of the number of matched nuggets to the total number of vital nuggets previously determined for that specific question. Practical nugget precision P_{def} is based on answer length $|a_i|$ (character-level or word-level) and approximates *true* nugget precision:

$$P_{\text{def}} = \begin{cases} 1 & if \quad |a_i| < L_a \\ 1 - L_a/|a_i| & \text{otherwise} \end{cases} \tag{4.6}$$

where L_a is the answer length allowance, which specifies the number of characters (or words) each definitional question's answer is allowed to cover. Answer length $|a_i|$ is the actual length of the answer a_i in terms of characters or word tokens, depending how the evaluation is performed. For definitional questions, the overall score was the harmonic average – F-measure – between nugget precision and nugget recall. In different years, definitional track performance assigned different values to the F-measure β parameter (i.e. $\beta = 3$ or $\beta = 5$):

$$F(\beta) = \frac{(\beta^2 + 1)P_{def}R_{def}}{\beta^2 P_{def} + R_{def}} \tag{4.7}$$

In the 2005 evaluation, the performance of the top system on definitional questions according to this metric was 0.248 F-measure$_{\beta=3}$ and a human manual run was 0.299 indicating that the human evaluation model does not entirely correspond to the definition of nugget F-measure, possibly due to the forced notion of nugget precision. However, nugget recall seems to be a more useful measure since humans can easily process a slightly more verbose answer as long as it contains the correct information – e.g. *he was the scheeming and ruthless sheriff of Nottingham* vs. *sheriff of Nottingham*. Under IBQA, answers are represented by short phrases, and therefore recall-based scores are more relevant than nugget-based NIST scores.

CHAPTER 5

Question Clustering

Contributions: *We introduce a data driven paradigm under which training questions are clustered rather than matched against an answer type set or against an ontology. Question clustering has the advantage of being less domain specific and thus more portable than predefined ontologies. It also allows multiple granularity clusters (types) with various degrees of overlap.*

Traditional pipeline question answering systems have a question analysis component which classifies the question according to a question ontology, extracts question keywords, and applies various types processing to questions (e.g. parsing, named entity extraction etc). The question clustering component of an instance-based question answering approach is equivalent to the question analysis component in these QA systems. Under the instance-based approach, training questions are clustered according to different similarity criteria into multiple clusters. Based on the assumption that each cluster contains a set of similar training questions, we derive cluster-specific answering strategies and apply them to new, test questions

The goal of the question clustering is to obtain a more flexible, domain-independent set of answer types as a basis for defining answering strategies, compared to pre-determined, disjoint question ontologies. During **training** the inputs to the question clustering component are simple questions (note that we do not perform clustering using the answers) and the output is a clustering of the training questions that include a measure of cluster quality. After question clustering, cluster-specific answering strategies are learned. During **testing**, the new question is represented in the same space used for clustering and relevant clusters are identified. The answering strategies corresponding only to relevant clusters are activated to seek correct answers to the new question. This is similar to using question ontologies, only instead of single, pre-defined ontology nodes, IBQA employs multiple, data-generated clusters.

When multiple clusters of various granularities and sizes are generated, multiple answering strategies are automatically constructed and since they were trained on different data, they offer a higher method diversity for obtaining correct answers. This applies especially when new test questions can be classified into more than one cluster and therefore several answering strategies are simultaneously activated.

For each cluster of training questions under the instance-based approach, we estimate cluster quality, based on features such as cluster size, cluster cohesiveness, and cluster granularity, as well as the probability of generating a successful answering strategy, relative to a new test question.

In the first part of this chapter we provide an overview of how question clustering fits into the IBQA framework. We discuss various methods of clustering and their advantages and disadvantages to the instance-based framework. We also present various methods for measuring cluster quality, similarity metrics, and clustering criteria that are possible under the IBQA framework. Starting with section 5.5 we present actual *implementation* choices, and component-level experiments, and results.

5.1 Related Work

Machine learning techniques for question classification are often based on a small set of answer types (e.g. location, height, person) most of which can easily be generated using a named entity tagger. These answer types are usually arranged in a shallow taxonomy of two or three levels which were initially created with the purpose of named entity tagging [9, 113].

The task of classifying questions according to these simple taxonomies adapted for QA has been successfully approached ($\sim 90\%$ accuracy) using machine learning techniques such as: hierarchical classifiers [64] and support vector machine classifiers [39, 136]. However, for more comprehensive ontologies, manual labeling becomes much harder and time consuming, and is likely to require expert labeling for domain-specific question and answer ontologies. Larger QA taxonomies [51, 50, 52] of around 100 nodes which combine answer types and question types based on user intentions and semantic roles have been built, but the component performance has not been independently analyzed. Question have been classified according to these taxonomies using a few hundred hand-written rules [44].

Finer-grained ontologies have also been used to classify questions. Harabagiu et. all [42] links subtrees in WordNet to labels given by a named entity tagger and in order to recognize a more detailed answer type. Similarly, Mann [78] builds a proper name ontology from free text and uses it to improve QA performance on proper name related questions. In the vast clustering literature, particularly relevant are the expectation maximization (EM) algorithm [28], as well as hierarchical clustering approaches and online clustering methods [29, 43]. Clustering criteria ranging from sum of squared errors to graph theoretic methods provide insight into estimating cluster quality for already defined clusters.

5.2 Assessing Cluster Quality

Faced with a new question, an instance-based QA system needs to find relevant clusters of training data (question-answer pairs) from which successful strategies can be generated and learned. While the overall task of learning an answering strategy from a set of questions

Test Question: *What is the longest river in the U.S.?*

ID	*Size*	*Cluster Description*
$C1$	103	What is <**NounPhrase**> in <**NounPhrase**>?
$C2$	2	What is the longest <**NounPhrase**> in the U.S.?
$C3$	3	Name the <**Superlative**> <**NounPhrase**> in the U.S.?
$C4$	12	What is the <**Superlative**> <**NounPhrase**> in the U.S.?
$C5$	24	What is the <**Superlative**> <**NounPhrase**> in the world?
$C6$	26	What is the <**Superlative**> <**NounPhrase**> in <**NounPhrase**>?
		. . .

Table 5.1: Examples of clusters covering training questions similar to the test question. For each cluster an answering strategy is learned from its questions. The answering strategy is then applied to the test question in order to identify potential answers. A confidence score is associated with each potential answer.

could be regarded as non-trivial and overwhelming, an initial concern is how to measure cluster relevance with respect to the new question, what aspects are good predictors of a cluster likely to generate successful strategies, and what features are conducive to robust learning. Moreover, what methods of clustering are most appropriate for this natural language application?

We approach the question clustering problem by first identifying important factors that influence the quality and viability of a cluster in the process of answering a new question. In order to better illustrate the necessity of incorporating these features into an instance-based approach to question answering, we provide examples associated with the scenario described in Table 5.1.

1. **Cluster Size**: in order to be able to learn how to answer similar questions, we need sufficient training data. Are there enough data points to statistically support the results?

   Example: assume that answering strategies learned from clusters $C3$ and $C4$ produce answers with equal confidence scores for the same test question. Clearly, since cluster $C4$ contains more training questions, it is more likely that its answering strategy be more robust than the strategy learned from $C3$, which covers fewer training questions. We expect answer confidence estimates to be more accurate when the training set size has more data points. Therefore, when computing the overall answer confidence, an instance-based QA system can benefit from incorporating a measure of cluster size as an indicator of of cluster quality. The more training data points in a cluster, the more accurate the confidence estimate.

2. **Cluster Relevance** to the test question: the likelihood that the new question can be answered by strategies learned from a particular cluster. Intuitively, this element measures the similarity between the test question and a cluster.

   Example: assume that answering strategies learned from clusters $C4$ and $C5$ produce answers with equal confidence scores for the same test question. In our example, the test question is more similar to cluster $C4$ than it is to cluster $C5$. Clearly, given an appropriate distance metric, the more similar a question is to a cluster, the more likely it is for that cluster to generate relevant strategies. Based on this observation, it is beneficial to account for cluster relevance in the overall answer confidence.

3. **Cluster Granularity**: corresponds to the specificity of a cluster – how similar the training questions in the cluster are. A measure of granularity the notion of *diameter* of a cluster. The smaller the diameter, the more similar the questions are and the more focused the answering strategy is likely to be.

   Example: assume that a good answering strategy cannot be learned from cluster $C2$ since it is too narrow since there are not enough training questions. Increasing the granularity of the cluster leads to clusters $C4$ and $C6$ which include increasingly more training data and are more likely to generate answering strategies that can answer new questions.

   At the same time, cluster $C1$ covers too many distinct questions from which it is difficult to learn a single successful answering strategy. Decreasing the granularity also leads to clusters $C6$ and $C4$ which are more focused (exhibit less variation in the training data) and are more likely to generate answering strategies that can answer new questions

   It is useful to consider cluster granularity when comparing different answers. A narrow cluster with sufficient data is more likely to generate high confidence answers than a more inclusive one.

4. **Cluster Scatter**: corresponds to the distribution of data points in a cluster. Given a fixed cluster definition, the more diverse the training questions, the more generalizable will the answering strategy be.

   Example: consider cluster $C4$. Assume that nearly all of the training questions have the superlative term "fastest". The answering strategy learned from such a cluster is less likely to generalize and answer new questions with different superlative terms. However, if the training questions in $C4$ cover many different superlatives, a better cluster strategy can be learned for that cluster. Hence, given a fixed cluster granularity, a uniform scatter in this constrained space is desirable in order to ensure a good generalization.

5. **Domain Constraints**: clusters must have a minimum number of training questions to ensure learning, questions in a clusters must have a minimal structure in common (e.g. have the same wh-word in common), questions in a cluster must share a minimum number of words, etc.

   Example: consider cluster $C2$. Although it is very specific and very similar to the test question, a constraint on the minimum number of data points required in a cluster prevents it from being a viable cluster for learning an answering strategy.

All of these factors could be beneficial when taken into account for estimating the quality of a cluster with respect to answering a new question. Most QA systems, in the initial stages of the question answering process, classify new questions into a set of pre-determined classes and apply corresponding carefully defined and tuned answering strategies. An instance-based QA approach generates several clusters of *similar* questions of different granularity, size, scatter, and relevance, and uses them to automatically learn answering strategies. This approach has the advantage of searching for correct answers using very different strategies based on very different types of questions in parallel. The cluster choices do not only determine which answering strategies are constructed, but they also directly influence overall individual answer confidences.

5.3 Clustering Paradigms

In clustering training questions, it is necessary to observe the above mentioned properties and requirements. We investigate the compatibility of several clustering paradigms and their potential application to the clustering of questions.

Features	Q_1	Q_2	Q_3	Q_4	$\cdots$
Words $\supset$ {“*where*”}	1	0	1	0	$\cdots$
Words $\supset$ {“*Clinton*”}	0	1	0	0	$\cdots$
Words $\supset$ {“*discovered*”}	0	0	1	0	$\cdots$
$\cdots$					
Pattern: {“*where is* $< NP >$ *produced*”}	1	0	0	0	$\cdots$
Pattern: {“*where is* $< NP >< VB >$”}	1	0	0	0	$\cdots$
Pattern: {“*where was <Proper Name> born*”}	0	0	0	0	$\cdots$
$\cdots$					
Answer Type: {*weight*}	0	1	0	1	$\cdots$
Answer Type: {*area*}	0	0	0	1	$\cdots$
$\cdots$					

Table 5.2: Features are extracted from questions and used as dimensions in a multi-dimensional space. These features can be words, patterns, answer types, n-grams etc and depend on the type of processing available (e.g. part of speech tagging). In this table Q_i are training questions and the binary features take values depending on the presence or absence of the features in the question.

The first step in clustering questions is to represent them as points in a multi-dimensional space. Towards this goal, we first extract features from the questions and then use these features as dimensions of the representation space (Table 5.2). The features that can be extracted are: lexical items (words – e.g. *who, killed, John, F., Kennedy*), n-grams (e.g. *who killed, John F. Kennedy, etc,* generic patterns that include pre-processing information such as part of speech tagging (e.g. *who* $< VB >< NP >$), answer types (e.g. weight, area) etc (Figure 5.1). Any clustering algorithm based on features extracted from the questions can be used under the IBQA framework. Different implementations may choose different algorithms, features, and pre-processing. This section explores various clustering paradigms and evaluates the benefits and limitations of each method with respect to question clustering.

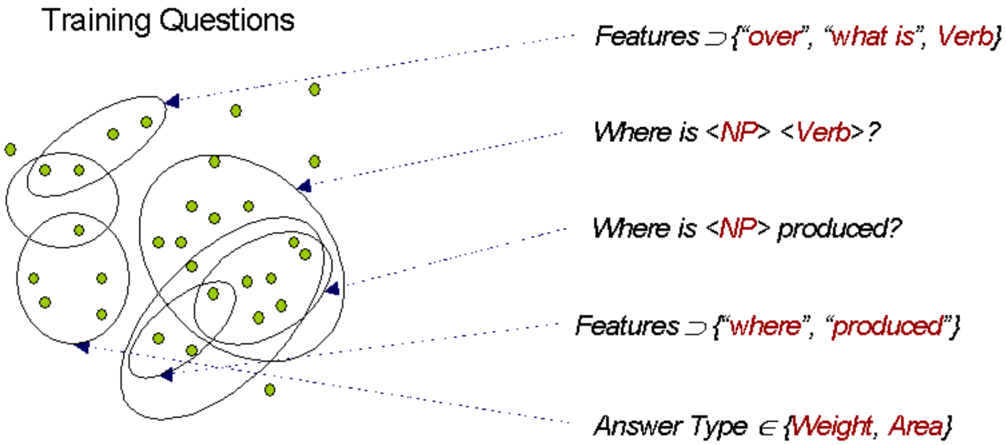
Training Questions
Features ⊃ {"over", "what is", Verb}
Where is <NP> <Verb>?
Where is <NP> produced?
Features ⊃ {"where", "produced"}
Answer Type ∈ {Weight, Area}

Figure 5.1: Examples of training question clusters – different clusters are based on different similarity criteria: the existence of certain words, the existence of certain patterns, or part of speech elements.

It is useful to note that the vector space in which we represent questions can be defined in multiple ways. The number of clusters required for different types of questions is unknown and not fixed. The form of cluster conditioned probability densities are unknown, and it is unlikely that the corresponding unknown covariance matrices are identical across clusters. Moreover, not all optimization criteria are meaningful in the question domain.

Since we desire different granularity clusters as well as different coverages of the question space, any type of cluster overlap or inclusion is acceptable and even desirable.

5.3.1 Iterative Optimization Clustering Algorithms

The expectation-maximization (EM) algorithm is a general method for estimating a set of unknown parameters θ that describe an underlying probability distribution, given the observed data D_o produced by this distribution. The EM algorithm searches for the maximum likelihood hypothesis (which consists of the values of the unknown parameters) by iteratively seeking the hypothesis h that maximizes the expected value of the log-likelihood of the data D given a hypothesis. More formally, the EM algorithm iterates over the following two steps:

Expectation Step: at j^{th} step, compute the $Q(h'|h)$ function, where h is the current hypothesis and h' is the new hypothesis:

$$Q(h'|h) \leftarrow E[lnP(D|h')|h, D_o] \tag{5.1}$$

Maximization Step: replace hypothesis h by new hypothesis h' that maximizes the Q function:

$$h \leftarrow \underset{h'}{\operatorname{argmax}}\ Q(h'|h) \tag{5.2}$$

In the case of question answering, the observed data is composed of the training questions represented as points in a multi-dimensional space. The un-observed data consists of the means and variances of the underlying distributions. In our experiments we used Euclidean distance as our distance metric for clustering (through the Weka machine learning toolkit [129]).

Since we are interested in clustering the training questions, it is natural to view the latent component of the data as the centroids (or means) of the clusters. The *soft* K-means algorithm, also called *fuzzy k-means*, is an application of EM for estimating the means of a mixture of k distributions. In this setting, the maximum likelihood hypothesis minimizes a weighted sum of squared errors.

In the question clustering setting, given a particular question dataset, it is not known whether the underlying distributions are gaussians or not. If we assume gaussian distributions, we must estimate the number of clusters as well as the set of unknown parameters (e.g. μ_i, σ_i), and we cannot assume that the covariance matrices are identical. For many applications, the number of clusters k (components in the mixture) is unknown. The choice of k is usually highly dependent on the task and nature of the data. There are several statistical methods (e.g. the gap statistic [119]) for estimating k and many are use the within-cluster similarity measure. The underlying assumption is that as the within-cluster similarity increases, the less likely it is to have large clusters that contain more than one “natural cluster”. When clustering questions, it is not clear what the “natural state” of the data is in terms of granularity, cluster overlap, and how representative of the *true* question space is the training set. Even if an acceptable solution for k is found, most of the time it will not cover the set of all *meaningful* clusters.

5.3.2 Combinatorial Clustering Algorithms

Combinatorial clustering algorithms are based on the assumption that one can assign data points to clusters without taking into account the underlying probabilistic model describing the data. The algorithm directly employs a loss function and attempts to minimize it through a combinatorial optimization algorithm. A very intuitive loss function is the within-cluster point scatter $W(C)$:

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(j)=k} d(x_i, x_j) \tag{5.3}$$

where $C(i) = k$ is a mapping (or *encoder*) function between a data point x_i and a cluster k, and $d(x_i, x_j)$ is a distance metric between two data points x_i and x_j.

Attempting to minimize the within-cluster point scatter loss function is equivalent to maximize the between-cluster point scatter [43], which is an intuitive process. However this loss function assumes that in question clustering we desire well-isolated and well defined clusters across the whole data. While it is useful to identify high density question clusters, it is also useful to consider looser and less well-defined groups questions which have commonalities.

Combinatorial optimization algorithms are by nature applicable on very small data sets and with a small number of clusters. Practical algorithms based on this principle are forced to cover only a small part of the problem space and perform variations of iterative gradient descent. Because of this, these algorithms are only guaranteed to converge to local optima. Although small, the question answering data is still too large for a practical application of full optimization algorithms. Even with good heuristics and different starting points, clustering in the question space would cover too little data (small parts in the encoder function space) in order to produce a large number of meaningful clusters. In terms of estimating the number of clusters, the same set of statistical and heuristic methods can be applied. Consequently, the same drawbacks and incompatibilities exist with respect to clustering in the question domain for the question answering task.

5.3.3 Hierarchical Clustering

As we have seen in previous examples, the task of clustering questions for QA requires the generation of different granularity clusters. Hierarchical clustering algorithms come closer to meeting that goal because they inherently assume the notion of multiple level structured data. The main component of a hierarchical clustering algorithm is defining a distance metric (similarity) between disjoint groups of data points, based on an individual pairwise distance measure.

Bottom-up, *agglomerative clustering* algorithms start with a single data point in each cluster. Then, they recursively merge pairs of clusters with the highest similarity into a single cluster. Very often between-cluster similarity metrics are used to decide what levels in the hierarchy most likely match the "natural" clustered state of the data. In question clustering, since more than one granularity is desirable, a large part of the hierarchy at different levels may contain desirable clusters of training questions.

The most frequently used methods of agglomerative clustering are *single linkage* (SL) which represents cluster similarity through the most similar data pair, *complete linkage* (CL) which represents cluster similarity through the most dissimilar data pair, and *group average* (GA) which uses the average pairwise similarity between two clusters. Usually, the three methods produce very different hierarchies: they tend to agree only when data exhibits strong clustering tendency (highly within-cluster similarity). When clustering questions, the strong clustering property is not always present throughout the question space.

Top-down, *divisive clustering* methods start with all the points into a cluster and recursively divide clusters based on highest between-partition dissimilarity. At each level, any flat clustering method (such as K-means) with $k = 2$ can be applied in order to split set of data points. However, a more principled divisive method [74] splits a cluster based on a grouping with the lowest similarity from the rest of the data.

Hierarchical clustering methods inherently incorporate the notion of granularity, which is necessary for the QA task. It also avoids the dependency on guessing the number of clusters and the starting configuration since only two clusters are merged at a time and the clustering extremes are one data point per cluster and all data points in a cluster.

However, different hierarchical clustering methods can lead very different data structuring (hierarchies). The outcomes are very also very sensitive to small changes in data. Moreover, these algorithms assume and impose a hierarchical structure on the data, which is not always desirable when working with questions: cluster overlaps are very frequent and do not always reflect the inclusion relation. Additionally, these methods are designed to find *one* best hierarchy, closest to the "natural structure" of the data. In question classification, different segmentations may reflect different question structures and therefore different equally viable hierarchies.

5.3.4 Constrained Subset Generation

In constrained subset generation, the idea is to generate all possible clusters (subsets of the training data) that obey a certain set of *constraints*. In some cases a *prototype* is given at run-time and the clustering criterion (subset generation criterion) is partially defined using the prototype. This has the effect of filtering the data to a much smaller neighborhood, then generating all possible clustering that meet a set of constraints. The clustering problem can be viewed as finding the k nearest neighbors according to a set of constraints, then clustering the neighborhood.

Clearly, in the worst case scenario given a generic cluster setting, this method is equivalent to generating the superset of the training data set (Algorithm 1). If there are no constraints, this approach is not feasible since the number of clusters that can be generated is $N!$, where N is the number of questions in the training data. The set of viable clusters **V** generated from prototype x_π, constraints **c** and training data $x_1 \mathinner{..} x_N$ is:

$$\mathbf{V}(\mathbf{x}, x_\pi, \mathbf{c}) = \{C \subseteq \{x_1, x_2, \mathinner{..}, x_N\} \ \mid \ c_i(C, x_\pi) = 1, \ \forall \ i = 0 \mathinner{..} m \ \} \tag{5.4}$$

Algorithm 1: Constraint Subset Generation

Require: processed training questions

```
for all training questions $Q_i$ do
    for number of gaps 0 .. MaxNumGaps do
        for all gap sizes and gap combinations do
            select gap starting index, and gap size
            if valid gap combination then
                generate pattern; store pattern
            end if
        end for
    end for
end for

for all patterns $p$ do
    for all constraints $c$ do
        test constraint $c$: e.g. minimum number of questions with pattern $p$
        if constraint un-matched then
            remove pattern from pool
            Simplest clustering: generate a cluster from pattern
            Dimension generation: treat pattern as a feature/dimension
        end if
    end for
end for
```

Note: the algorithm can be implemented efficiently, especially if question size is under 100 words. The more constraints are used, the fewer clusters are generated.

$$\text{where } c_i(C, x_\pi) = \begin{cases} 1 & \text{if the } i^{th} \text{ constraint is met} \\ 0 & \text{otherwise} \end{cases}$$

However, in practice under very restrictive constraints and with either small dimensionality or with a small set of non-zero features (data points have non-zero components along few dimensions), this approach becomes feasible. In question clustering, question length average is under ten words and the set of features that describe a question is very small even when using n-grams, part of speech tags, and parsing components,. Under restrictive constraints (e.g. constraints on: minimum cluster size, minimum number of feature shared within a

Prototype: *What is the longest river in the U.S.?*

ID	*Size*	*Viable*	*Cluster Description*
C1	30	yes	What is <**NounPhrase**> in <**NounPhrase**>?
C2	20	yes	What is <**NounPhrase**> in <**ProperNoun**>?
C3	20	yes	What is the <**NounPhrase**> in the <**ProperNoun**>?
C4	154	yes	What <**QTerm**> in <**QTerm**>?
C5	2	no	What is <**QTerm**> river in <**QTerm**>?
C6	0	no	What is <**QTerm**> river <**QTerm**> U.S.?
qC7	0	no	<**QTerm**> is the longest river <**QTerm**>?
C8	560	yes	What is <**QTerm**>?
			. . .

Table 5.3: Example of question clustering using a *prototype* and set of loose *constraints*: the minimum cluster size is 3 and clusters must share at least 2 surface tokens with the prototype). The surface tokens in this case are the actual words that are not abstracted as more complex features: e.g. words such as "*what*", "*is*", "*longest*", and "*river*". The number of tokens in a question is small and the pre-processing is usually limited (part of speech, noun-phrase identification, parsing, named entity tagging) and is a function of question length. Therefore, it is feasible to generate possible clusters using prototype and constraints, and then populate the viable clusters with training data.

cluster, minimum number of features shared to a test "prototype" question) the number of clusters generated is manageable. For example, it is practical to restrict the training clusters to sharing a minimum surface form with the test question and have each cluster contain a minimum of h data points.

The constrained subset generation setting is different but related to *leader-follower learning* in which new patterns are presented online to an existing set of clusters and the centroid closest to the pattern is altered to incorporate it. In the constrained subset generation, only a small set of possible subsets of various granularities and overlaps are *activated* when new patterns are being presented.

By providing a prototype and defining constraints for the data within clusters as well as constraints related to the prototype, the cluster space is greatly reduced. Since this process is application specific and highly depends on the data set size, dimensionality, and prototype (if defined), no upper bounds on the number of clusters can be derived.

Some generated clusters may not be meaningful (i.e. do not match a "natural state" of the data), yet may still obey the constraints. In this case, training questions covered by these clusters do not share aspects useful in finding answers to similar questions. Hence, they are less likely to generate useful strategies that lead to the correct answer. The hope is that bad clusters are not able to generalize and to lead to good answering strategies and subsequently to high confidence answers. If they do, then the questions in such clusters actually share meaningful features. For example, cluster $C4$ (Table 5.3) matches all the constraints, but is not likely to generalize from training data and produce high confidence answers to a similar question.

The constrained subset generation approach benefits from multiple granularity and no restrictions or assumptions on a fixed number of clusters that best matches the data. Moreover, there is no structure such as a hierarchy imposed on the clusters and any type of overlap is permitted. The constraints serve as an application-specific viability filter for the clusters generated. The downside of this approach is that it is less stable as a generic clustering method and the application-specific notion of success is highly dependent on the constraints.

5.4 Similarity Metrics & Clustering Criteria

The simplest and most frequently used similarity metric is the Euclidean distance, or more generally the *Minknowski metric*, which reduces to the *Euclidean distance* when $q = 2$ and to the *Manhattan distance* when $q = 1$.

$$\delta(\mathbf{x}, \mathbf{x}') = \left(\sum_{\mathbf{k=1}}^{\mathbf{D}} |\mathbf{x_k} - \mathbf{x'_k}|^{\mathbf{q}} \right)^{\frac{\mathbf{1}}{\mathbf{q}}} \tag{5.5}$$

Choosing the Euclidean distance as a measure of (lack of) similarity has an impact on the clustering outcome. This choice assumes that the feature space is somewhat isotropic and all directions are equally important. Data is not invariant to transformations that distort distance relationships: e.g. scaling of axes. Data normalization is usually undesirable since it often reduces the *distance* between clusters.

The *cosine similarity metric* is another way to characterize the similarity between two questions, which is invariant to rotation and dilation but not invariant to translation or other linear transformations:

$$cos(\mathbf{Q_1}, \mathbf{Q_2}) = \frac{\mathbf{Q_1} \mathbf{Q_2}}{||\mathbf{Q_1}|| \cdot ||\mathbf{Q_2}||} \tag{5.6}$$

Using this metric is based on the assumption that the angle between two questions in the training data vector space is meaningful and the feature space is well defined. In question clustering, many features can be defined as binary-valued attributes (e.g. *question contains n-gram "what is"*). In this case, the cosine metric measures the commonality of features between questions. Similar metrics are based on the fraction of features shared and *Tanimoto distance* (ratio of attributes shared to the number of distinct features).

Another type of similarity metric is based on word-level *edit distance* between questions. The basic idea is to compute the smallest number of insertions, deletions, and substitutions required to change one string into another. More generally, the operators allowed for transforming a string into another can be weighed differently according to the cost of applying that operator. This problem is also called pairwise alignment since by using deletions and insertions, we compute the cost of aligning two sequences of tokens. Dynamic programming solutions [24] are very fast, especially for short strings – which is the case in the question domain. The more similar questions are at a word level, the smaller the cost for *editing* one question to obtain another. Various types of substitutions can be defined as a function of properties of the words or phrases to be substituted.

The *BLAST* family of algorithms is a class of dynamic programming algorithms related to edit distance and was designed for fast searching in nucleotide and protein databases. BLAST focuses on regions of *local* alignment in order to detect relationships among sequences which share only isolated regions of similarity [4]. Sequence alignment is used to compare new sequences with previously characterized genes. However, since question datasets are currently several orders of magnitude less than protein databases, full alignment can be computed very fast.

Another class of similarity metrics is based on tree structure. If questions are processed with syntactic parsers, tree-structures become associated with each data-point in a cluster. Therefore, it is natural to define similarity metrics based on tree properties. The notion of similarity could reflect: the longest path in the tree that two questions share, tree depth, number of nodes at each depth, size of the longest common sub-tree etc.

A clustering criterion offers a quantitative way of evaluating the quality of clusters. In a typical clustering problem, a clustering criterion is defined and then clusters are generated by optimizing that criterion. In the constrained subset generation setting, clusters are generated and then one or more metrics are required to evaluate independently the quality of each cluster. We take advantage of different clustering criteria and use them as individual measures of cluster quality in the context of instance-based question answering.

The *sum of squared error* criterion J_s measures the average deviation from the cluster mean m_i and is usually used as a criterion that generates k clusters of minimum variance:

$$J_s = \sum_{j=1}^{k} \sum_{Q \in C_j} ||Q - m_j||^2 \tag{5.7}$$

In the case of subset generation, we are interested in adapting this criterion to measure the sum of squared error for individual clusters:

$$J'_s(C_j) = \sum_{Q \in C_j} ||Q - m_j||^2 \tag{5.8}$$

This is exactly R_j, which we use as a measure of cluster granularity. Related to the J_s criterion, is a criterion which attempts to optimize the average squared distance between points in a cluster:

$$J_p = \frac{1}{|C_j|^2} \cdot \sum_{Q_a \in C_j} \sum_{Q_b \in C_j} ||Q_a - Q_b||^2 \tag{5.9}$$

This criterion has the advantage of considering the relative distribution of points in a cluster. We have used this criterion as a the basis for the point scatter in an individual cluster S_j. Two additional clustering criteria that are frequently used are simply the minimum and maximum distance between pairs of points in a cluster.

$$J_M(C_j) = \max_{Q_i, Q'_i \in C_j} \delta(Q_i, Q'_i) \quad (5.10)$$

$$J_m(C_j) = \min_{Q_i, Q'_i \in C_j} \delta(Q_i, Q'_i) \quad (5.11)$$

Although more simplistic, they are sometimes a better match to the nature of the data, thus producing better clusters.

5.5 Question Clustering in IBQA

The question clustering task in the instance-based question answering approach benefits from low dimensionality of data (processed question) and small training dataset size (limited number of questions). Given a new test question the instance-based approach generates clusters of training questions according to a particular clustering method. If we view the test question as a prototype and the language independent constraints on question similarity as application specific clustering constraints, the *constrained subset generation* method becomes a natural choice for question clustering. Using very few simple constraints, it generates a limited number of training data clusters of varying granularity and overlap.

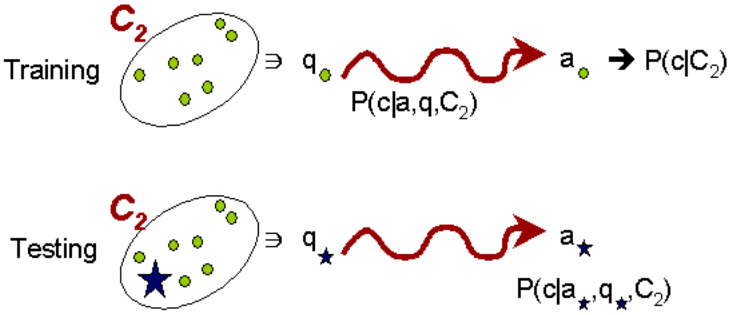

Figure 5.2: Cluster-level training and testing: During training an answering strategy is learned from training questions in each cluster. During testing the answering strategy is applied to test questions that are also covered by the same cluster.

Algorithm 2: Question Clustering: Training

Require: training questions

```
for all questions do
    pre-process question – e.g. POS tagging, NE tagging
    collect features from question: words, n-grams, patterns
end for
filter features by frequency and constraints
define multi-dimensional space using features
represent training questions as a vector of features
cluster questions (e.g. CSG using our implementation)
for all clusters do
    filter cluster according to constraints (e.g. minimum # questions)
    if cluster viable then
        learn answering strategy from cluster
    end if
end for
```

Algorithm 3: Question Clustering: Testing

Require: clusters of training questions, test question q

```
for all clusters $C_i$ do
    if test quetion $q$ is in cluster $C_i$ then
        estimate cluster $C_i$ relevance $R(C_i, q)$ to $q$
        apply answering strategy specific to $C_i$ learned during training to question $q$
        use $R(C_i, q)$ as a weight for all extracted answers
    end if
end for
```

During the training process (Figure 5.2), the training questions are clustered and for each cluster, individual answering strategies are learned (Algorithm 2). The cluster quality and strategy probability of success (obtaining correct answers) are estimated over all training questions in the cluster. During testing (Figure 5.2), we are given a new question and a set of relevant clusters (Algorithm 3). For each cluster we apply the corresponding answering strategy to the test question and we obtain an answer set. The probability of success is computed using the individual answering strategy the answers came from.

The questions are pre-processed using part-of-speech tagging and shallow parsing. Several additional surface-form features are extracted from the questions: acronyms, names, and punctuation. The questions are tokenized using text normalization – e.g. punctuation such as commas are separated from words, apostrophe 's' is separated to denote possessive, contractions are resolved etc. No reference resolution is performed within each question or across questions. In section 5.5.1 we discuss different question features we extracted in our IBQA implementation.

Although it is a very simple method, the constrained subset generation method has the advantage of being able to generate several different clusterings according to different similarity metrics. IT also allows the models derived from data to reflect the quality of the cluster in their performance. Clusters that cover too few data points (training questions) as well as clusters which cover too much of the training data are not able to consistently generate models that yield correct answers on similar training questions and are therefore assigned low confidence scores. We explore several types of clustering methods previously mentioned and investigate whether they are able to produce most of the same high confidence clusters, without generating a prohibitive number of noisy clusters.

5.5.1 Extracting Features for Question Clustering

The first step in performing question clustering is to determine the feature space used to define the questions. Question clustering is performed on the set of training questions projected onto this multi-dimensional space. In this space, resulting clusters are used as more focused training bases for learning individual answering strategies. In question clustering, there are several features that can be extracted and used regardless of the domain or language (e.g. surface form features, capitalization) and there are also features that are language and resource dependent (e.g. part of speech tagging, parsing, named entity extraction). The instance-based framework is defined irrespective of the features or the clustering method. In our implementation of the instance-based approach, we have implemented the following features:

- *surface form features* – the simplest types of features are based on the actual tokens (words) found in questions. More specifically, we identify n-grams that are shared among the training questions and collect them only if a minimum number of questions include them. In our implementation we used a minimum of 3 shared words per cluster.

- *capitalization* – starting with the previously extracted n-grams, proper names, titles, acronyms are identified and tagged accordingly, to be further used by retrieval and extraction.

- *classing* – language independent classing includes digit classing – e.g. the number 15 is represented as a double digit token $\bar{d}\bar{d}$ – and number classing – e.g. the number 15 can be represented as number token $\bar{n}$). Language dependent classing includes frequently occurring classes of words such as months (January, Dec.) and weekdays (Monday, Tue.)

- *part of speech tagging* – although language specific, noun and verb identification is a very important part of question analysis and therefore a feature for similarity among questions in question clustering.

- *morphing* – a morphological analyzer is able to provide much of the mapping between different forms related to the same word: e.g. parts of speech, conjugations, number.

- *named entity tagging* – identification of named entities such as people, organizations, and locations in the text of questions is a very strong indicator of similarity going beyond surface-form and providing the equivalent of basic semantic categories for questions.

- *corpus and question statistics* – basic statistics about the question and the corpus from which answers are to be extracted may provide useful, albeit simple insights. For example question length may be a primitive statement that very long sentences are less likely to be in clusters with high similarity.

Part of speech tagging and can also describe simple features that can be extracted questions. More complex features consist of n-grams of the above features: e.g. n-grams of actual tokens, n-grams of part of speech tags etc. The tradeoff between language dependency and flexibility of features to describe questions is evident. We plan to incorporate these features incrementally into the question clustering component of our implementation of the IBQA approach, in order to better understand this tradeoff.

A centroid of a question cluster is defined in this context simply as the centroid of all data points representing projections of questions onto this space. It also incorporates the constraints imposed on the particular question cluster it represents: e.g. must contain a WH-word (i.e. when, who, where etc). In the following subsections we explore various issues in generating question clusters and evaluating their quality.

5.5.2 Estimating Cluster Quality

We estimate the usefulness/quality of a cluster C_j containing training questions $\mathbf{Q}$ with respect to a test question q by taking into consideration cluster size, relevance, and cohesion. We present here a simple local model for combining cluster properties and constraints in order to estimate the quality of a cluster in the context of a new question. Section 9.2 further explores additional overall models for combining individual quality estimates from each stage (e.g. clustering, retrieval) in the instance-based question answering process.

We view the radius of a particular cluster C_j as the average distance between each data point Q_i and the centroid. The larger the radius of a cluster, the less specific it is, and the less similar training questions in the cluster are to new questions. We measure the *radius* R_j of a cluster by averaging the distances to the centroid:

$$R_j = \frac{1}{|C_j|} \cdot \sum_{i=0}^{N} b(Q_i, C_j) \cdot \delta_c(Q_i, C_j) \tag{5.12}$$

where $b(Q_i, C_j)$ is a $\{0, 1\}$ valued function that signals the presence of training question Q_i in cluster C_j and $\delta_c(q, C_j)$ is a distance metric between a question q and the centroid of cluster C_j.

By measuring the scatter within a particular cluster S_j, we essentially measure how well the training data covers the space around the centroid. This is equivalent to estimating how likely it is for a cluster to generalize to new questions. We measure the scatter of data points in a specific cluster by examining the normalized sum of the distances between pairs of training questions in a cluster:

$$S_j = \frac{1}{|C_j|^2} \cdot \frac{1}{2R_j} \cdot \sum_{Q_i \in C_j} \sum_{Q_l \in C_j} \delta_c(Q_i, Q_l) \tag{5.13}$$

where $1/|C_j|^2$ is a normalization factor over the number of data points in the cluster, $1/2R_j$ is a normalization factor using a measure of cluster diameter, and $\delta_c(Q_i, Q_l)$ is a distance metric between two questions.

The size of a cluster is strong indicator of how successful the answering strategies based on the local cluster data will be. It is also a measure of the confidence these strategies should have in a proposed answer. The more local training data in a cluster, the more confident the strategies should be in proposing a candidate answer. We choose to estimate the quality of a cluster by incorporating a function of cluster size:

$$s_j = 1 - e^{-\frac{|C_j|}{\beta R_j}} \tag{5.14}$$

where the β parameter controls when there are a sufficient number of questions in the cluster – i.e. when there are enough data points to produce confident strategies. The function also varies with the radius of a cluster since answer confidence varies with how well the cluster space is covered by the training data.

The set of application-specific constraints **c** that are imposed on clusters to ensure viability V can be viewed as a factor in estimating the quality of a cluster:

$$V_j(q) = \prod_{i=0}^{I} c_i(C, q) \tag{5.15}$$

where

$$c_i(C, q) = \begin{cases} 1 & \text{if the } i^{th} \text{ constraint is met} \\ 0 & \text{otherwise} \end{cases}$$

and where the constraints are defined as a function of the cluster C_j and of the test question q, considered to be the prototype.

In an instance-based question answering approach, we need to estimate the likelihood that strategies learned from a cluster of similar training questions will generalize to a new test question. We define the quality $Q(C_j, q)$ of a cluster C_j given the q test question q as:

$$Q(C_j, q) = \frac{s_j \cdot S_j}{R_j} \cdot \frac{1}{\delta_c(q, C_j)} \cdot V_j(q) \tag{5.16}$$

The factors that depend only on the training data are functions of cluster size, scatter, and radius. The factors that take into account the test question are the relevance (distance) of the cluster to the new question as well as the viability of a cluster in the context of the test question.

Using the expressions in (5.12), (5.13), (5.14), (5.15), we can re-state equation (5.16) as:

$$\begin{aligned} Q(C_j, q) \;=\; & \frac{\left(1 - e^{-\frac{|C_j|}{\beta R_j}}\right) \cdot \left(\sum_{Q_i \in C_j} \sum_{Q_l \in C_j} \delta_c(Q_i, Q_l)\right)}{2 \cdot |C_j| \cdot \sum_{i=0}^{N} b(Q_i, C_j) \cdot \delta_c(Q_i, C_j)} \\ & \cdot \frac{1}{\delta_c(q, C_j)} \cdot \prod_{i=0}^{I} c_i(C, q) \end{aligned} \tag{5.17}$$

The cluster quality measure is one of the factors used in the candidate answer correctness estimation. By making a soft decision on the quality of a cluster, we allow multiple strategies of different granularity, relevance, and strength to be generated from similar training questions. Section 5.4 describes several similarity metrics in the question domain that can be used to define the dissimilarity (distance) function δ_c and it covers several widely used clustering criteria.

5.6 Question Clustering Experiments

The first clustering method we have employed is constraint subset generation (CSG). As constraints for clustering we have imposed a minimum of three training questions per cluster, a minimum of three word overlap among questions in a cluster, a maximum gap size of four words, and a minimum number of gaps of four. This method over-generates clusters. However, by estimating cluster quality and relevance, the weights associated with the cluster-specific answering strategies are very low. We have also imposed a threshold on the cluster quality estimate – implementation dependent. As an alternative to the constrained subset generation clustering, we also used the expectation-maximization algorithm (EM) as implemented in the Weka toolkit [129] using the default parameters: (100 iterations, minimum standard deviation of 10E-6, and no cluster number specification). We applied EM hierarchically, using several constraints for the indivisibility property of a cluster (e.g. cardinality). We trained the hierarchical EM clustering models on the same features used for the CSG clustering: surface-form (n-grams, paraphrases, POS tags etc). Table 5.4 compares the performance of an instance-based system when the two clustering methods are used. It also shows the difference in the number of clusters produced. One major difference between the two methods is that the hierarchical EM algorithm only produces overlapping clusters when they have an inclusion relation.

	MRR	Top5	# clusters
CSG	0.432	0.496	906
EM	0.272	0.322	241

Table 5.4: **Mean Reciprocal Rank (MRR) and Correct in TopK (Top5)** scores for the instance-based system (proximity extractor with answer merging) using constrained subset generation clustering (CSG) versus hierarchical expectation maximization (EM) clustering. We show MRR/Top5 performance as well as the number of clusters (answering strategies) generated.

We have also experimented with the Cobweb hierarchical clustering algorithm and with the K-means algorithm implemented in the Weka toolkit [129]. Variations of Cobweb parameters (acuity and cutoff) were yielding very unstable clustering results. The Weka K-means implementation requires the user to pre-specify the number of clusters. We also opted not to use k-means since for a hierarchical clustering, too many parameters would need to be specified. However, there are methods [83, 118, 30] designed for automatically estimating the number of clusters to be used. Although the using EM yields a lower overall IBQA performance, the number of answering strategies generated is much lower than using the CSG clustering method. Thus the former method may be beneficial in highly interactive scenarios, where fewer strategies should be activated. In future sections, we also present an answering strategy selection method that can reduce the number of strategies activated, but maintain a relatively high performance.

5.7 Question Clustering – Summary

In this chapter we introduced a data driven paradigm under which training questions are clustered as opposed to matched against an answer type set or against an ontology. Compared to predefined ontologies, question clustering is more domain and question-set independent and thus more portable and adaptable. With question clustering, multiple granularity clusters are considered to be question types. The clusters have various degrees of overlap, which helps represent the training question datasets better.

We have presented the general question clustering problems and compared it with question clustering. Towards clustering in the IBQA framework, we discussed several possible clustering paradigms, distance metrics, and clustering criteria. In terms of implementation, we have used two clustering methods: CSG and hierarchical EM (as implemented by the Weka data mining toolkit), and in terms of distance metrics, we have used Eucledian distance and cosine similarity. Specific to our implementation, we have shown feature extraction from questions, used to define vector space dimensionality and we have also shown how we estimate cluster quality, as well of cluster relevance to a test question. Combined, these two cluster-quality estimates are used as weights for the answering strategies that are learned (see following chapters) from individual clusters – i.e. each answer extracted using the cluster-specific answering strategy is weighed using the cluster quality/relevance estimate.

For the component level experiments we have compared hierarchical EM with CSG clustering. CSG clustering performs better due to the *weighted* over-generation of clusters. The downside with this method is the amount of processing involved in generating the large number clusters during training and applying them during testing. As model reduction alternative, EM reduces the number of clusters considerably, and also obtains a lower performance. A better implementation of the hierarchical part of EM that accounts for overlapping clusters has the potential to bridge the performance gap with CSG.

CHAPTER 6

Answer Modeling

Contributions: *In terms of answer modeling for QA, this thesis proposes modeling the expected answer as a distribution over answer types as opposed to rigidly using a single answer type. The answer type distribution is built directly from a local cluster of training questions.*

The goal of the answer type modeling element in a question answering system is to estimate the semantic type of the expected answer, given as input the raw question. The answer type is very useful and often critical in subsequent stages in the QA process. Based on the expected answer type: i) document retrieval can be better guided (via query type selection or query content) to potentially retrieve more relevant documents, ii) answer extraction can identify sentences containing words or phrases that fit the expected answer type profile, thus restricting answer choices and focusing extraction to likely candidates, and finally iii) answer merging methods can be tailored to the specific answer type – i.e. date answers are merged differently than person name answers.

In this chapter we first present general answer type modeling issues, introduce clustering as a viable method for answer modeling, comparing it to using answer ontologies. We then discuss our implementation of answer modeling in IBQA and present component level experiments and results.

Several approaches for answer type modeling have been developed, mostly focusing on obtaining a single very specific answer type for every question. Most existing question answering systems classify new questions according to static ontologies [14, 50, 52] and take the answer type to be the classification output – i.e. the ontology node. These ontologies incorporate knowledge about the expected answer (e.g. date, location, person), answer type granularity (e.g. date, year, century), and very often include semantic information about the question type type – e.g. birth date, discovery date, death date. The question type contains additional semantic information that cannot be inferred independently from a correct answer. For example from the correct answer *January 27, 1756*, we can only know that the answer type could be a date. However, given the question *When was Mozart born?*, we can narrow down the answer type more specifically to a date of birth.

While effective to some degree, many of these ontologies are still very small, and inconsistent. Considerable effort has been invested into building and maintaining increasingly accurate and fine-grained ontologies. Semantic classes can also be extracted from hierarchical resources such as WordNet [82] to form the basis for automatically constructing ontologies. Viewed from this perspective, the expected answer type corresponds to a distinct category in a semantic network such as WordNet. However, answer types are arguably not always disjoint and hierarchical in nature. For example, the question "*Where is the corpus callosum?*" expects an answer that could be considered both location and body part. In many reasonable ontologies, these two concepts would constitute different nodes, most likely not in a parent-child relation.

A significant drawback to using ontologies is question answering systems do not follow a standardized ontology, making individual component evaluation very difficult and re-training for new question datasets time-consuming. Moreover, very often, systems use their own QA system-specific answer type ontology, adding to the complexity of reproducing the same results.

The task of determining the answer type of a question is usually considered a *hard decision*[1] problem: questions are classified according to an answer ontology. The classification (location, person's name, etc) is usually performed in the beginning of the QA process and all subsequent efforts are focused on finding answers of that particular type. Several existing QA systems implement feedback loops [42] or full-fledged planning [93, 48] to allow for potential answer type re-classification.

Very often, questions can have multiple correct answers belonging to different answer types. These answer types can have partial or no overlap ,and may also have various degrees of granularity. Collectively these answer types can be thought of as forming the basis for an answer type *distribution*. To illustrate this point, all three questions in 6.1 can accommodate answers of types: *full date*, *year*, and *decade*.

Question	**Answer**
When did Glen lift off in Friendship7?	Feb. 20, 1962
When did Glen join NASA?	1959
When did Glen have long hair?	the fifties

Table 6.1: Questions whose expected answer type is temporal; however, note that the granularity of the answer type varies across questions.

However, it can be argued that *full date* is the most likely answer type to be observed for the first question, *year* the most likely type for the second question, and *decade* the most likely type for the third question. In fact, although the three questions can be answered by various temporal expressions, their answer type distributions can be quite different. Existing answer models do not usually account for these distributions, even though there is a clear potential for better answer extraction and more refined answer scoring.

A more flexible choice is to model answer type – equivalent to the semantic class – distributions for individual questions, based on known answer types observed in similar training questions. This approach has the potential of finding a more accurate set of expected answer types for more ambiguous questions. We show that answer type detection performance can

[1] the answer is classified into a single class instead of generating a probability distribution over answers

be improved by taking into account the similarity of new questions to training questions. This approach has the potential of improving answer modeling performance by directly incorporating answer type distributions into statistical QA systems.

6.1 Related Work

In most QA systems, the first step towards finding answers is question analysis. During this step, questions are classified and assigned semantic tags. These semantic tags usually come from named entity tags, hand-crafted ontologies [14, 50], or semantic classes in WordNet. Very often, after performing question analysis, QA systems make a hard decision on what the expected answer type should be and proceed accordingly.

Large corpora such as the Web can be mined for simple patterns [106] corresponding to individual question types. These patterns are then applied to test questions in order to extract answers. Other methods [31] rely solely on answer redundancy: high performance retrieval engines and large corpora contribute to the fact that the most redundant entity is very often the correct answer.

Different classes of questions can be answered by different answer types, whose prior likelihoods are not necessarily equal. By making *hard* decisions concerning the answer type, systems are likely to be overly selective and reject good answers because they do not match an expected answer type. An average mutual information model [77] for question classes and semantic tags achieves 0.4 MRR for trivia questions whose answers are considered to be one word extracted from web documents. In these experiments named entity tags [14] were better suited than WordNet [82] classes at representing semantic tags.

The IBM statistical question answering system [55, 56, 18] uses maximum entropy to model answer correctness by introducing a hidden variable representing the expected answer type. The expected answer type is defined as one of five major named entity types, and as an approximation, only the most likely entity type predicted by an answer tag model is considered. Using this entity type, the probability of an answer being correct is computed.

Another answer extraction approach [79] models the answer correctness based on semantic tags associated with WH-words (e.g. When, Where, Who, What) and with keywords (anchors) that appear both in the question and in the answer context. Three simple features based on proximity of anchors to answers are also used to improve performance. The approach is especially useful when the answer class is clearly stated: e.g. *"What is the color of sapphires?"* or *"Which country borders Belize?"*, making this statistical approach more conducive to finding good semantic classes for answers. For these questions, answer types can be identified using semantic ontologies and exact matching. When answers are proper nouns, mutual information is used to associate named entity tags from known answer contexts with specific question WH-words (e.g. Who → {Person, Location, etc}). However, this model does not incorporate a similarity measure between test questions and training questions other than matching WH-words.

6.2 Answer Modeling under IBQA

The answer model is the first step in an IBQA cluster-specific answering strategy (Figure 6.1). Under the instance-based question answering, we address the sub-task of answer modeling through learning cluster-specific answer type distributions and using a set of most likely answer types during retrieval and extraction stages. The answer types are given by the thousands of synsets found in WordNet and in order to find expected answer types we have implemented both a traditional classifier (SVM) approach as well as a k-nearest neighbor algorithm to generate cluster-based answer type distributions. Below, we first motivate the necessity of soft decisions for answer types and we describe our algorithm.

Learning specific answer type distributions is useful not only in terms of identifying answers in running text but also in terms of answer ranking. A probabilistic approach has the advantage of being able to postpone answer type decisions from early in the QA process until the answer extraction or answer ranking stages. Instead of selecting the wrong answer type with a low error margin, the expected answer type distribution can be used by subsequent stages in the QA process. A probabilistic approach also has the advantage of allowing

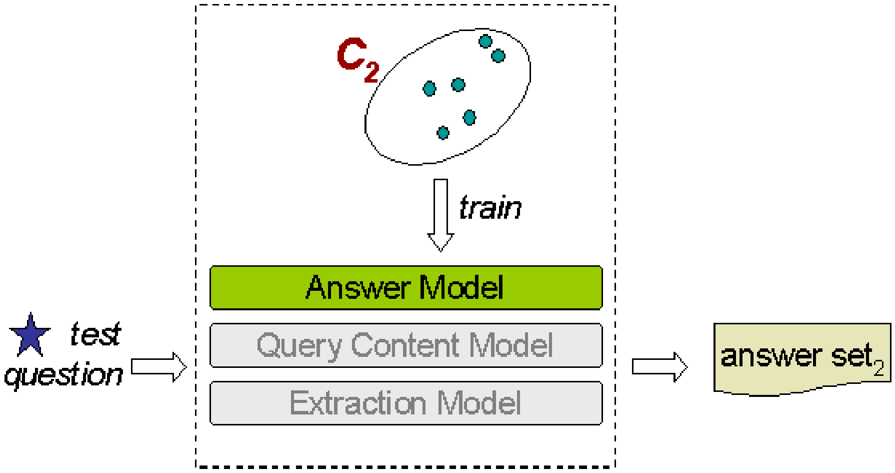

Figure 6.1: Answer modeling as the first component of an answering strategy.

different training data to shape the expected answer structure and type differently.

Who <**Verb**> <**NP**> . . . ?	**Person**	**Country**	**Animal**	**Other**
Who killed Kennedy?	0.85	0.08	0.01	0.06
Who attacked France in 1940?	0.17	0.74	0.00	0.09
Who saved Timmy when he fell into the well?	0.71	0.01	0.22	0.06
Cluster answer type distribution	0.58	0.28	0.08	0.06

Table 6.2: Answer type distribution in a cluster can be derived directly from answer type distributions for individual training questions.

The answer modeling task consists of learning specific answer type distributions for individual test questions from other similar training questions. Table 6.2 shows that answer type distributions can be very different for questions that would normally be classified into the same class (e.g. *proper name* or *person*). In particular the third distribution assigns more probability mass on *animal* being the answer type. These values can be estimated from large amounts of text data (e.g. local corpora or the web) using simple techniques. Under the instance-based approach, the simplest method for estimating these types is by pooling together the answer types of all instances in a cluster. Although it is not within the scope of this work, more complex techniques could be employed that take into account the lexical, syntactic, and semantic features of individual test questions and combining them with the cluster-based evidence.

Very often, question analysis components generates ambiguous or overly generic classifications. For example, simple question classifiers may assign the following classes to the questions shown in Table 6.2: *proper name*, *who-question* or *person name*. The former two classes are too broad and the latter class is too specific. Such classes often correspond to named entity tags or nodes in a hand-built ontology.

6.2.1 Generating Answer Type Distributions

Due to the probabilistic nature of the instance-based approach, answer type distributions can have a significant impact in the way answer types are considered throughout a QA system. We have used this method as the main answer modeling approach in our experiments.

As mentioned above, a very simple but general method for learning distributions of answer types is to consider similar training questions and combine their answer type distributions. Since many QA systems already employ named entity tags or question taxonomies, similarity could be defined as questions that belong to the same class (e.g. *proper name*). However, this type of similarity is not sufficient for determining more fine-grained answer types that may incorporate semantic information.

Under a *local uniform* approach, all training questions in the same class as the test question contribute equally to the new answer type distribution. A more flexible approach relies on question similarity to adjust the contribution of each training question to the test question's expected answer type distribution. If a test question q is more similar to a training question Q_i than to another training question Q_j, than the relative weight of the answer type contribution of Q_i to the expected answer type of question q will be higher than the weight of the answer type contribution of Q_j.

Using training questions and their corresponding known correct answers, we can approximate the distribution of expected answer type for each test question. We estimate the probability $P(\alpha_t|q, C_j)$ of observing an answer of type α_t when asking a question from class C_j as:

$$P(\alpha_t|q, C_j) = \mu \cdot \sum_{Q_i \in C_j} P(\alpha_t|Q_i) \cdot \delta_a(q, Q_i) \tag{6.1}$$

where $P(\alpha_t|Q_i)$ is the probability of observing an answer of type α_t when asking a question Q_i, $\delta_a(q, Q_i)$ represents a distance function between two questions q and Q_i, and μ is a normalization factor over the set of viable answer types in class C_j. Note that if $\delta_a(q, Q_i)$ is always 1, we obtain the local uniform approach under which all questions in the same class (e.g. Who-questions or Proper Name questions) contribute equally to the final answer type distribution. For $\delta_a(q, Q_i)$, we have experimented with cosine similarity and Euclidean distance, using the same multi-dimensional space the training questions are represented in.

A training question Q_i has one or more corresponding known correct answers $a_{i1}..a_{ik}$. Therefore, the probability of a question observing an answer of type $P(\alpha_t|Q_i)$ can be expressed as:

$$P(\alpha_t|Q_i) \quad = \quad \frac{1}{|\mathbf{A_i}|} \cdot \sum_{j=0}^{|\mathbf{A_i}|} P(\alpha_t|a_{ij}) \tag{6.2}$$

where $\mathbf{A_i}$ is the set of known correct answers corresponding to training question Q_i, and a_{ij} is the j^{th} element of that set. The probability $P(\alpha_t|a_{ij})$ of an answer type α_t given an actual answer a_{ij} is very often 1 for a particular answer type and 0 for other answer types. However, this is not always true. For example answer “New York” sometimes refers to “New York City” and sometimes refers to “the state of New York”.

When estimating the probability of an answer type given a cluster, equation (6.1) can be generalized by incorporating the answer type distribution outside of the class of interest C_j as additional information into a simple model:

$$P(\alpha_t|q, C_j) = \lambda \cdot \left[\mu_1 \cdot \sum_{Q_i \in C_j} P(\alpha_t|Q_i) \cdot \delta_a(q, Q_i)\right] + (1-\lambda) \cdot \left[\mu_2 \cdot \sum_{Q_i \notin C_j} P(\alpha_t|Q_i) \cdot \delta_a(q, Q_i)\right] \tag{6.3}$$

where λ is the parameter that adjusts the contribution of the out-of-cluster answer type distribution. If a question taxonomy is used, equation 6.3 can be generalized further to a mixture model that incorporates answer distributions of super-classes (parent nodes in the taxonomy):

$$P(\alpha_t|q, C_1) = \lambda_1 \cdot \left[\mu_1 \cdot \sum_{Q_i \in C_1} P(\alpha_t|Q_i) \cdot \delta_a(q, Q_i)\right] \cdots + \lambda_k \cdot \left[\mu_k \cdot \sum_{Q_i \in C_k} P(\alpha_t|Q_i) \cdot \delta_a(q, Q_i)\right] \tag{6.4}$$

where $C_1 \subset \cdots \subset C_k$ are subclasses in a question answering taxonomy and $\lambda_1 \ldots \lambda_2$ are mixture parameters that can be trained in order to optimize a specific criterion (e.g. distribution divergence).

6.2.2 The Nature of Answer Types

In previous sections we have discussed about answer type distributions and we have shown how to estimate the probability of a new answer type given a very generic class of training questions, without specifying what an answer type really is. Under our IBQA approach, the definition of answer type is flexible, depending on the resources available. We mention below several dimensions that can be used to define an answer type.

The simplest, resource-free approach is to use surface form features and text structure features in order to help define answer types: e.g. answer contains a sequence of tokens beginning with an uppercase letter; answer does not contain commas; digit patterns (‘$\bar{d}\bar{d}$) / ($\bar{d}\bar{d}$) / ($\bar{d}\bar{d}\bar{d}\bar{d}$) where $\bar{d}$ signifies a digit. Digit classing, case information, and text structure are also successfully used in other tasks such as named entity tagging [9]. However, the expressiveness of surface-form alone is limited and while it can cover temporal and numeric expressions, and partially proper names, it cannot generalize well for other answer types.

Part of speech taggers and syntactic parsers can be used to better define the structure of the answer (e.g. proper noun, noun phrase). Answers to factoid questions are usually noun phrases and have specific contextual and structural features corresponding to syntactic elements. Named entity taggers are commonly used in QA systems for answer typing. Typically, these sets are very small and restrictive, but they provide a very broad and intuitive answer type classification.

Semantic categories can also be defined as possible answer types. WordNet classes have been previously used in order to specify the type of answer that must be produced by a QA system. This approach has the advantage of providing a large and versatile answer type set, as well as being already organized into an (hypernymy) ontology. Shallower ontologies [50, 53] ranging from fifty to a few hundred nodes have also been constructed from named entity tags.

When measuring the contribution of a training question Q_i to the answer type distribution for a test question q, the distance $\delta_a(q, Q_i)$ reflects the similarity between Q_i and q. Similarity metrics such as Euclidean distance, cosine similarity, and edit distance, as described in section 5.4 can also be applied to measure the dissimilarity between individual questions.

6.3 Experiments & Results

The first experiment tests the hypothesis that using more than one answer type is helpful. We are focusing on granularity as the difference between answer types: i.e. city vs. country. For this experiment, we used questions with *location* answer type from past TREC evaluations. Our test set includes 160 location type questions, most of which have “Where” as the the WH-word. Answers were manually annotated using a set of semantic tags from the extended QA ontology of the Javelin question answering system [93]: country, city, state, county, region, state, park, cemetery, structure, body part, object, etc. We compared the true distribution of correct answers with a rigid estimate and a soft answer type distribution. Very often, correct answers have only one granularity level: e.g. country. However, sometimes correct answers may have different granularities: e.g. both country and city can be valid types.

When estimating the similarity between questions, we computed an edit distance that places more importance on prepositions, conjunctions, and proper names. Better similarity functions may be obtained by incorporating into the distance metric question parse tree information. After computing the distance between a test question q and a training question Q, we used a decaying exponential function $e^{-\alpha \cdot \delta_a(q,Q)}$ as a weighting function (based on the edit distance previously computed) to Q's contribution for the expected answer type distribution of q. In the *local uniform* case $\alpha = 0$, which means that all training questions contribute equally with answer types to the expected answer type of the test question. $alpha$ can be set differently according to the question datasets, answer types considered etc. We tuned $alpha$ for this experiment using five development questions.

We compared the true distribution (TrueD) of each test question with the generated answer type distribution using *Kullback-Leibler divergence* (KL) which is the relative entropy between two distributions, *Jensen-Shannon divergence* (JS) which is just the symmetrized KL-divergence, *cosine similarity*, and the χ^2 *statistic* which measures the likelihood of one distribution being drawn from another. We performed leave-one-out cross validation experiments to generate answer type distributions for individual questions. In this experiment we first build the UnifD (uniform weights) answer type distribution by simply using the frequency of occurrence of each answer type in all similar questions. The SimD distribution is built using a KNN approach: each training question's answer type(s) to the expected answer type distribution with a weight proportional to the inverse distance between the test question and the training question.

Due to the relatively small number of answer types (under 30), the cosine similarity criterion was the most stable – performed well, with smallest variations across sentences. When using the average cosine similarity across all test questions we obtained an overall similarity score of (0.56) between the true distribution and the generated distribution. By using the cosine similarity metric, all other metrics – i.e. the values of JS, KL, and χ^2 – were also close to their minima.

Figure 6.2 shows examples of true and generated distributions for several questions. In most cases, the distribution of expected answer type based on question similarity is much closer to the true distribution than the uniform average distribution.

Intuitively, the more training data is available, the more likely it is for test questions to find similar training questions that also have similar answer type distributions. Figure 6.3 confirms this intuition: the more data we use for training, the higher the cosine similarity measure between the true and generated distributions. Similarly performance was obtained with the KL-divergence, JS-divergence, and the χ^2 measures decrease with more data, indicating a smaller divergence in the true and generated distributions.

One of the advantages of generating answer type distributions for new questions is adaptability to available training data. Known correct answers reflect closely the corpus from which they were extracted. Because of this, expected answer type distributions are also specific to specific corpora used during training. However, given sufficient training questions, the gap between true answer type distributions and generated expected answer distributions decreases, improving on a single, hard decision on answer type selection.

Correctness Classification

Before we experiment with answer type distributions, we implemented another existing approach to answer modeling: casting it as a classification problem. Under this approach, each question is classified into several classes corresponding to specific answer types – e.g. location, date. The answers are often classified in a set of named entities (e.g. person name, organization, location), a shallow answer type ontology, or a more sophisticated ontology based on resources such as WordNet. In our experiments we use the nodes in the WordNet hierarchical semantic network as answer type classes. We performed leave-one-out cross validation by training support vector machine classifiers [121]. In particular we used the SVM Light toolkit [58] with a linear kernel and the corresponding default parameters. The task is of the classifier is to predict answer types from features derived from individual questions (data points).

The features we used in these experiments are: lexical items (actual words), answer type overlap measures computed as the percentage of the answer type explicitly found in the question, named entities (e.g. presence or absence of person names), WordNet synsets (hypernym hierarchy nodes) and bigrams of lexical items. All features are represented as binary values, except answer type overlap which is represented as a floating point numbers.

Additive Feature Classes	**Micro Average**	**Macro Average**
Lexical unigrams	0.610	0.145
AType Overlap	0.635	0.158
Named Entities	0.710	0.172
WordNet	0.830	0.406
Bigrams	0.838	0.428

Table 6.3: Answer type classification – we train SVM classifiers based on several features: lexical unigrams, answer type overlap, named entities, WordNet and lexical bigrams. The micro and macro averages are shown for models based on cumulative (current and previously mentioned) features.

The set of labels for question classification consists of two kinds of answer types: WordNet nodes and generic classes. The generic classes consist of: proper name, date, and numeric, all of which are easy to identify. Table 6.4 shows the additive effect of successive feature classes to answer type classification performance. The baseline consists of a classifier trained on lexical features (i.e. the words in the question). We compute the overall classification accuracy, which in this case is the same as the Micro Average. In question answering, measuring micro-average is more relevant for the overall performance of the QA system. However, when performing error analysis, it is beneficial to consider the performance of individual classes (answer types). We present macro-average as an additional measure, although we are not trying to optimize for it. To the baseline we add lexical overlap with specific answer types, named entities, wordnet features classes, and finally extend the feature set to lexical bigrams (i.e. pairs of question words).

We have also experimented with slight variations of these features – for example, we have observed that including function words (e.g. of, for, in, and) in lexical unigram and bigram features improves performance. Also preserving case information is beneficial, especially for frequent answer types. The macro-average lower performance is due to the many classes we have considered: general classes as well as WordNet-based answer types. Since the classifier usually has lower performance on many small classes and the macro-average assigns equal weight to each class, the overall macro-average score is lower.

Sample Answer Types	**Recall**
Age	0.80
Location	0.92
Date	0.96
Distance	0.62
Time Interval	0.33
Speed	0.38
Acronym	0.71
Definitional	0.84

Table 6.4: Examples of question classification performance for several answer types.

In Table 6.4 we give some examples of answer types and corresponding performance in terms of recall: the fraction of questions of a particular answer type that were classified correctly. *Location* and *Date* perform well since both have specific structural and surface form characteristics such as capitalization (e.g. PA; Pittsburgh; December) and format (e.g. December $12^{th}, 2006$; Pittsburgh, PA). Although date have the advantage of more discriminative format, actual locations can be found in WordNet as hyponyms of nodes such as *location*, *country*, *peninsula*, *natural language* (e.g. French, Spanish, which are implicitly country indicators). However, the *Time Interval* and *Speed* answer types are more likely to be classified as a more general *Quantity* answer type. Another problem in answer type classification is that sometimes, very specific answer types corresponding to test questions are not found in the training data. In this cases, test questions are either misclassified or a more generic answer type is identified.

Among possible directions for extending answer type classification for high-performance

QA system is to enhance the question-based features. Stemming and morphological normalization could help since very often questions with the same answer type include different conjugations, number, or part-of-speech transformations. Furthermore, synonymy and hypernymy may help uncover deeper question similarity in terms of answer types. Finally these features can be combined and used as contiguous n-grams (e.g. "$\texttt{stem}(w_i)$, w_{i+1}, $\texttt{syn}(w_{i+2})$") or skip n-grams (e.g. "$\texttt{stem}(w_i)$, w_{i+2}, $\texttt{syn}(w_{i+4})$") to better capture contextual meaning.

Answer Type Distributions

When using a K-nearest neighbor approach to compute an expected answer type distribution for a test question, we can use the whole training question dataset as the set of neighbors. However, because of the vast differences among training questions, we decided to use KNN only based on training questions pertaining to each cluster (equation 6.4). This way, we ensure that we learn answer types from sets of already similar questions.

If no question classifiers are used for expected answer type modeling and no broad answer type classes are available (e.g. person, location, date etc), the IBQA approach offers an initial step towards generating expected answer type distributions by clustering the training questions [69]. For each cluster of similar training questions we generate a *true* answer type distribution – which is equivalent to using clusters instead of classes in equation 6.4. By using a KNN-style approach, we allow training questions to contribute differently to the expected answer type, based on their similarity to the test question.

The distribution divergence can be computed when using a QA ontology instead of the IBQA question clustering. Ideally, the generation of answer type distributions can be improved by closely coupling the influence of cluster-based training questions with an answer extraction model [77] that attempts to match semantic classes found in training questions with semantic classes found in sentences containing correct answers; however, this is not within the scope of this work. Since answer modeling is generally a separate stage in the question answering pipeline for most QA systems, expected answer type distributions can be incorporated into other statistical approaches [32, 56, 69].

When generating answer type distributions for individual clusters, an instance-based QA system is considering the answer types of training questions in that cluster. We are interested in how many of the most frequent answer types should be selected to obtain a good coverage of correct answers in test questions. Using our cluster-specific KNN model of the expected answer type, we obtain a coverage of correct answers from 93% of the questions.

Figure 6.4 shows the answer type distribution coverage of correct answers. By using WordNet classes for answer types, correct answer coverage increases significantly. By making a hard decision and using only one answer type for each question instance we obtain an answer type classification accuracy of 0.74. Just by adding an additional back-off expected answer type, we increase the answer type coverage to 0.85. Although subsequent answer types bring diminishing performance improvements, by using the top five most frequent answer types, we obtain almost maximum correct answer coverage, and considerably decrease the noise of larger clusters – larger clusters often include questions of different answer types.

6.4 Question Clustering – Summary

In this chapter we introduced a new method for modeling the expected answer as a distribution over answer types. This method is more flexible and yields good results when compared to more the traditional approach of rigidly using a single answer type. Rather than using the entire training dataset, answer type distributions are built directly from local clusters of training questions.

We have discussed the benefits of using an answer type distribution as the means of solving the problem of choosing a particular answer type granularity over another. By considering both granularities in the beginning of the QA process, IBQA carries the available information and avoids making a critical decision early on. A cluster-level answer type distribution is also a solution for the case where more than one answer types are being considered – no longer a simple granularity issue.

Through experiments we show that using more than one expected answer type is beneficial to the QA process by avoiding answer type misclassifications that thwart the success of answering strategies. We have described a cluster-specific KNN model of the expected answer type, and shown how to use training questions and their known correct answers to generate an answer type distribution. Using a traditional classification approach with a support vector machine classifier, we obtain a 0.838 accuracy and using the cluster-specific KNN approach, we obtain a 0.93 accuracy.

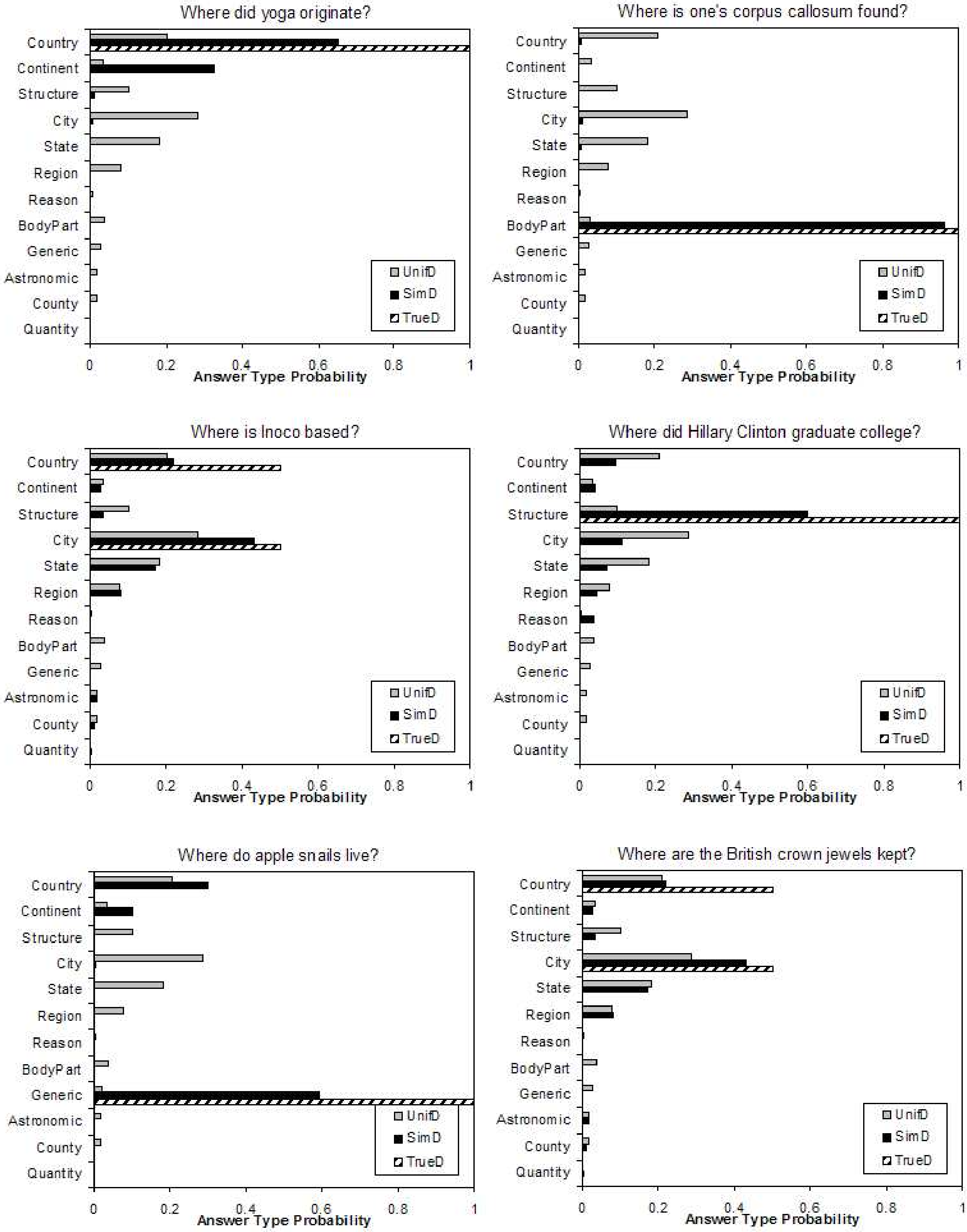

Figure 6.2: Answer Type distributions for several *location* questions. This figure shows the most common answer types. *TrueD* denotes the true distribution of answer types for each question, *SimD* denotes the distribution generated using a weighted (KNN) contribution from each training question, and *UnifD* denotes the distribution generated using uniform contributions from training questions.

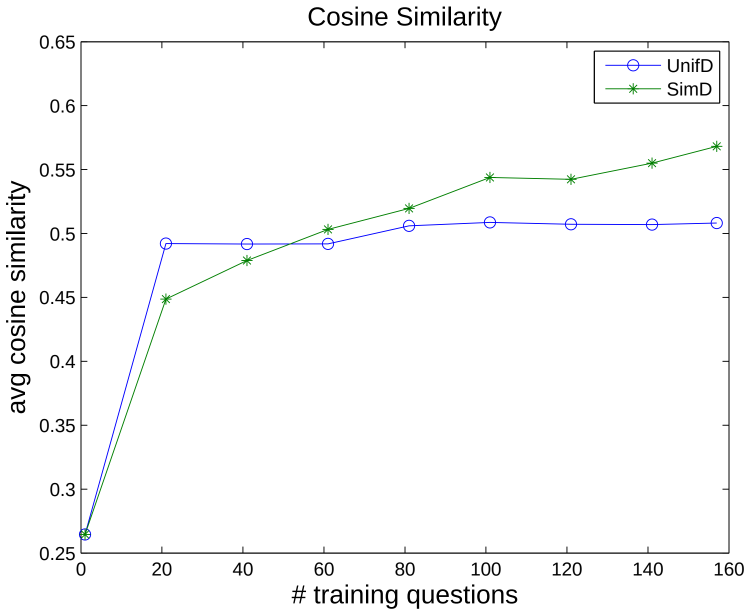

Figure 6.3: Comparison between an uniform contribution of answer types corresponding to training questions (UnifD) and a weighted contribution of answer types. Cosine similarity between the true and estimated answer type distributions increases with more training data for both methods. However, the weighted contribution scheme outperforms the uniform method.

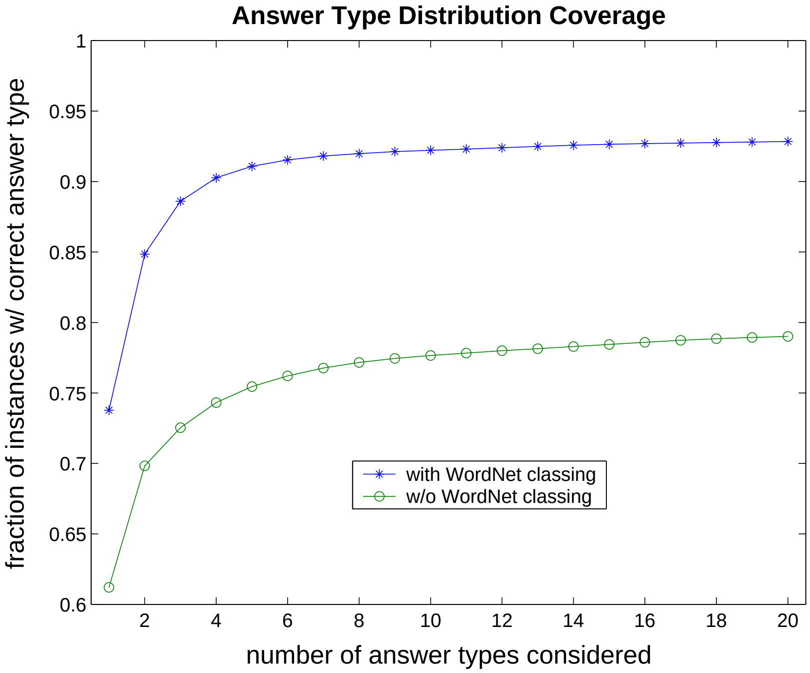

Figure 6.4: Answer type distribution coverage of correct answers: the fraction of question instances whose expected answer type distributions actually cover the correct answer type.

CHAPTER 7

Retrieval in Question Answering

Contributions: *In terms of retrieval for question answering, this thesis introduces a new, cluster-based query expansion method that learns queries which are successful on multiple similar questions. This method improves retrieval performance for QA when used in addition to existing query expansion methods.*

Given large datasets of raw text available, most question answering systems incorporate an information retrieval component in order to identify a set of relevant documents, likely to be on topic and to contain correct answers to questions. In question answering, document and passage retrieval operate under a slightly different information need than straight-forward document retrieval.

Document rank is still important in terms of how believable the source is, how much authority it holds, and its relevance to the topic at hand as expressed in the question and corresponding query. However, it is has less impact when it comes to answer extraction.

If a document contains an answer of the appropriate type, in the appropriate context, it is extracted regardless of the rank. While document/passage relevance to the query is still important, sometimes answers to questions appear in off-topic documents:

Question: *Where is London?*
Contexts: *… double deckers … be seen in London, England.*
… cheap flights to London, England.
… of dialing codes to London - England - UK …

Correct answer density in the top rank documents/passages is desired. From an answer extraction perspective, regardless of document topic, the more correct answers are present in raw text, the more likely it becomes for an information extractor to find a correct answer. It also follows that the higher the density, the more correct answers are likely to be extracted.

We define *relevance* of a document or a passage in a question answering setting: a piece of text is relevant if it contains a correct answer in a correct context. Since it is very difficult to automatically evaluate the correctness of context, notion of relevance is sometimes relaxed to whether a document contains the correct answer, regardless of context. Note that even if a document is on the same topic as the original question, if it does not contain a correct answer, it is still not considered directly relevant. However, it can still be used for query expansion.

As expected, the retrieval component must produce relevant documents with high density, simple contexts. However, it is not required to produce relevant documents from an IR point of view: i.e. documents can represent different topics, as long as they contain the required information. Topic relevance, and therefore rank is less important since documents must contain correct answers occurring in contexts conducive to information extraction – which is system dependent.

While precision is not an appropriate measure of document retrieval performance, a better and direct performance measure is the actual number of relevant documents retrieved. *Average precision*, *R-precision* and simply the *number of relevant documents retrieved* are also suitable retrieval metrics in a question answering context.

7.1 Related Work

Experiments [23] using the CMU Javelin [93] and Waterloo's MultiText [19] question answering systems corroborate the expected direct correlation between improved document retrieval performance and QA accuracy across systems. Effectiveness of the retrieval component was measured using *question coverage* – number of questions with at least one relevant document retrieved – and *mean average precision*. Results suggest that retrieval methods adapted for question answering which include question analysis performed better than adhoc IR methods which supports previous similar findings [90]. Another question answering study [89] explores the impact of document retrieval on the FlexIR vector space retrieval system and suggests that stemming leads to an improved performance.

In question answering context, queries are often ambiguous since they are directly derived from the question keywords. Such query ambiguity has been addressed in previous research [103] by extracting part of speech patterns and constructing clarification queries. Patterns are mapped into manually generated clarification questions and presented to the user. The results using the *clarity* [26] statistical measure suggest that query ambiguity is often reduced by using clarification queries which produce a more focused set of documents.

Another research direction that tailors the IR component to question answering systems focuses on query formulation and query expansion [130]. Taxonomic conceptual indexing system based on morphological, syntactic, and semantic features can be used to expand queries with inflected forms, hypernyms, and semantically related terms. In subsequent research [10], stemming is compared to query expansion using inflectional variants. On a particular question answering controled dataset, results show that expansion using inflectional variants produces higher recall than stemming. Terra and Clarke [116] study query expansion using lexical affinities with different query formulation strategies for passage retrieval. When evaluated on TREC datasets, their affinity replaement method obtained significant improvements in precision, but did not outperform other methods in terms of recall.

Within the context of extending the JAVELIN question answering system to restricted domains [95], the retrieval approach uses a successive relaxation of structured queries to retrieve relevant documents. The search is performed by searching for documents containing instances of predicates in raw text that match predicates in the question, that also contain mentions of the question entities. JAVELIN also extends the search through the use of WordNet and the CNS ontology.

Passage retrieval [20, 57, 55, 51, 108] is often preferred in question answering systems over document retrieval due to less raw text to be processing during extraction and sometimes reduced noise at the cost of coverage. A thorough evaluation of passage retrieval algorithms [115] shows that boolean querying schemes achieve good performance on the question answering task. Non-linear query term density functions for scoring passages tends to perform well. Another comparison between document and passage retrieval [22] shows that while passage retrieval has a lower coverage than document retrieval, it also reduces the amount of noise it passes to QA components that further process the passages.

Recent research shows that entity models for information retrieval [104] improve the performance of political orientation classification and of answering proper-name type questions (e.g. “Who is Powell?”, “What is IBM?”). In this framework, a language model (or word distribution) is associated with an entity (e.g. person, place, organization) and is then used in various tasks. Similar research [133] has independently focused on building entity profiles for definitional questions, by using various web-based structured and semi-structured resources, and then applying the profiles to local corpora in order to extract answers.

Predictive annotation [101] is one of the techniques that bring together corpus processing and smarter queries. Twenty classes of objects are identified and annotated in the corpus, and corresponding labels are used to enhance IR queries. Along the same lines, [3] propose a method for learning query transformations in order to improve answer retrieval. The method involves learning phrase features for question classification. [128] address the problem of query clustering based on semantic similarity and analyze several applications such as query re-formulation and index-term selection.

7.2 IBQA Approach to Retrieval

The Query Content Model is the second component (Figure 7.1) in a cluster-based answering strategy. Current question answering systems use IR in a straight-forward fashion: query terms are extracted and used to construct basic queries, which are later expanded using statistical methods, semantic and morphological processing. Documents are retrieved and the top K are further processed. The above approach describes the traditional IR task and does not take advantage of specific constraints, requirements, and rich context available in the QA process. Pseudo-relevance feedback (PRF) (figure 7.2) is often used in question answering, especially in web-based QA systems in order to improve the chances of retrieving relevant documents.

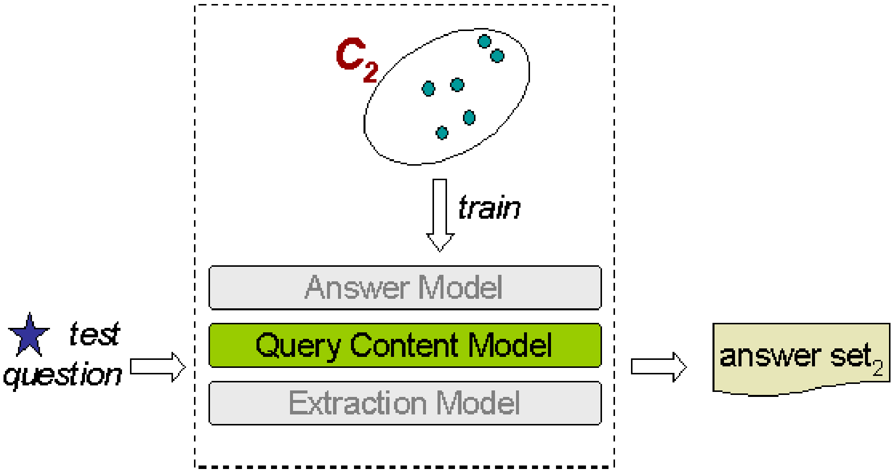

Figure 7.1: Query content modeling as the second component of an answering strategy.

Typical QA queries used in document or passage retrieval are constructed using morphological and semantic variations of the content words in the question. However, this type of queries does not benefit from the underlying structure of the question, nor does it benefit from available training data which provides similar questions that we already know how to answer.

In our IBQA framework, we introduce a new task-based method for query expansion that is complementary to existing strategies and that leads to *different* documents that contain

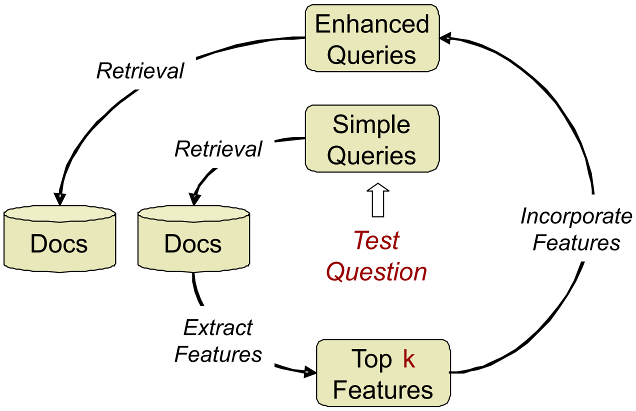

Figure 7.2: Pseudo-relevance feedback: at run-time, simple queries are generated from the test question, documents are retrieved and the most frequent content terms are used to enhance the simple queries. The enhanced queries are used to retrieve the final document set.

correct answers. Our approach goes beyond keyword-based methods and takes advantage of high-level correlations in the retrieval process for similar questions.

The central idea is to cluster available training questions and their known correct answers in order to exploit the commonality in the retrieval process. From each cluster of similar questions we learn a different, *shared* query content that is used in retrieving relevant documents - documents that contain correct answers. This method leverages the fact that answers to similar questions tend to share contextual features that can be used to enhance keyword-based queries. Experiments with question answering data show that our expanded queries include a different type of content compared to and in addition to existing methods. Since these queries have clusters as a source for expansion, we show they are conducive to the retrieval of *different* relevant documents.

The data-driven framework we propose takes advantage of knowledge available at retrieval time and incorporates it to create better cluster-specific queries. In addition to query expansion, the goal is to learn content features: n-grams and paraphrases [45, 54] which can be added to simple keyword-based queries in order to yield better results. We take advantage of the fact that for similar training questions, good IR queries are likely to share structure and content features. Such features can be learned from training data and then be applied to new similar questions. Note that some of these features cannot be generated through simple query expansion, which does not benefit from known similar queries.

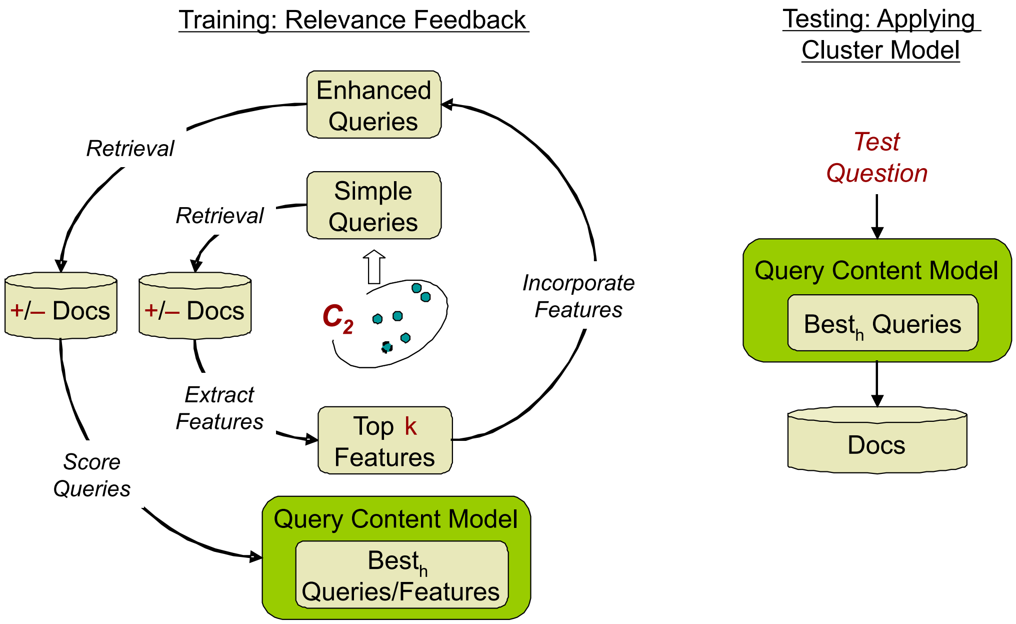

Figure 7.3: Cluster-based methods using task-based relevance feedback. During training, simple queries are generated from all questions in a cluster; corresponding document sets are retrieved; features most indicative of relevant documents are selected and used to construct enhanced queries and retrieve new document sets. Features that generate the best performing queries across an entire cluster are saved in the *Query Content Model*. These features are then used to construct specific queries for test questions.

Figure 7.3 shows how cluster-specific query content is learned. Notice that while PRF is performed on-line for each test question, relevance feedback is performed across all questions in each individual cluster. Relevance feedback is possible for training data, since correct answers are known and therefore document relevance can be automatically and accurately assessed.

A *Query Content Model* containing the resulting features (query types) is learned for each individual cluster. Algorithm 4 generates queries enhanced with cluster specific content, selects the best performing queries, and constructs the *Query Content Model* to be used online.

Initially, simple keyword queries are formulated using words and phrases extracted di-

Algorithm 4: Cluster-based relevance feedback algorithm for the retrieval component of the instance-based approach.

```
keywords/phrases are extracted from each training question
simple queries are built using only question keywords/phrases
**for all** simple queries **do**
    retrieve a set of documents
**end for**
documents are classified into relevant and non-relevant based on the presence or absence
  of (from training data) known correct answers
features (e.g. n-grams, paraphrases) are generated from all (cluster training questions')
  retrieved documents
feature selection (e.g. average mutual information) is performed top $k$ features most
  indicative of relevant documents are selected
enhanced queries are constructed by combining simple queries with the top $k$ features –
  adding one feature at a time ($k$ new queries)
**for all** enhanced queries **do**
    retrieve a set of documents
**end for**
documents are again classified into relevant and non-relevant based on the presence or
  absence of known (from training data) correct answers
enhanced queries are scored according to the density of relevant documents
the top $h$ features used in the previous step to construct enhanced queries that
performed best across all questions in the cluster are included in
the *Query Content Model* – up to $20$ queries in our implementation
```

rectly from the *free* question keywords that do not appear in the cluster definition. The keyword queries are then subjected to frequently used forms of query expansion such as inflectional variant expansion and semantic expansion (table 7.1). Further processing depends on the available and desired processing tools and can generate variations of the original queries: morphological analysis, part of speech tagging, syntactic parsing. Synonym expansion and corpus-based techniques can be employed as part of the query expansion process, which has been extensively studied [10].

Since most online search engine do not allow weighted term queries and limit the query size, the retrieval component of web-based question answering systems is drastically limited. However, when searching in a local corpus, expanded terms could have corresponding

When was the first postage stamp issued in the US?

keywords:	first AND postage AND stamp AND issued . . .
synonyms:	(first OR original OR initial) AND . . .
hypernyms:	. . . (issue OR distribute OR publicize) . . .
VB conj:	. . . (issue OR issued OR issuing OR issues) . . .
NN form:	. . . (stamp OR stamps) . . .

Table 7.1: Query expansion methods for question answering – query terms and query expanded terms according to noun forms, verb forms, hypernyms, and synonyms.

weights associated to them - e.g. hypernyms: “(1.0 issue) OR ($.7$ distribute) OR ($.6$ publicize)”.

Queries formulated using the methods mentioned above further benefit from additional search engine expansions using pseudo-relevance feedback. Such expansions are aimed at better recall at the expense of lower precision.

We introduce a new query expansion method that is cluster-based. It has the advantage of being orthogonal to traditional query expansion and can be used in addition to pseudo-relevance feedback. The cluster-based expansion is based on context shared by similar training questions in each cluster, rather than on individual question keywords. Since cluster-based expansion is based on different features compared to traditional expansion, the main benefit of the retrieval step in a QA system is that it brings in new relevant documents that are different from the ones retrieved using the existing expansion techniques.

7.3 Cluster-Based Query Expansion

Simple queries are run through a retrieval engine in order to produce a set of potentially relevant documents. While this step may produce relevant documents, we would like to construct more focused queries, likely to retrieve documents with correct answers and appropriate contexts. The goal is to add query content that increases retrieval performance on training questions. Towards this end, we evaluate the discriminative power of features (n-grams and paraphrases), and select the ones positively correlated with relevant documents and negatively correlated with non-relevant documents. This goal of this approach is to retrieve documents containing simple, high precision answer extraction patterns.

More specifically, consider a positive class consisting of documents which contain a correct answer, and a negative class consisting of documents which do not contain a correct answer. We compute the average mutual information[1] $I(C; F_i)$ between a class of a document, and the absence or presence of a feature f_i in the document [81]. We let C be the class variable and F_i the feature variable:

$$\begin{aligned} I(C; F_i) &= H(C) - H(C|F_i) \\ &= \sum_{c \in C} \sum_{f_i \in 0,1} P(c, f_i) \log \frac{P(c, f_i)}{P(c)P(f_i)} \end{aligned} \tag{7.1}$$

where $H(C)$ is the entropy of the class variable and $H(C|F_i)$ is the entropy of the class variable conditioned on the feature variable. Features that best discriminate passages containing correct answers from those that do not, are selected as potential candidates for enhancing keyword-based queries.

For each question-answer pair, we generate enhanced queries by individually adding selected features (e.g. Table 7.2) to simple queries. The resulting queries are subsequently run through a retrieval engine and scored using the measure of choice (e.g. average precision). The content features used to construct the top h features and corresponding enhanced queries are included in the *Query Content Model*.

7.3.1 Query Content Model

The *Query Content Model* is a collection of features used to enhance the content of queries which are successful across a range of similar questions (Table 7.2). The collection is *cluster specific* and not *instance specific*, meaning that features are derived from training data and

[1] as well as other statistics in section 7.4.1

Cluster: When did **X** start working for **Y**?

Simple Queries	*Query Content Model*
X, **Y**	"**X** joined **Y** in"
X, **Y** start working	"**X** started working for **Y**"
X, **Y** "start working"	"**X** was hired by **Y**"
...	"**Y** hired **X**"
	X, **Y** "job interview"
	...

Table 7.2: The Query Content Model is a *cluster-specific* collection of content features that generate the best document set. Queries based only on X and Y question terms may not be appropriate if the two entities share a long history. A focused, cluster-specific content model is likely to generate more precise queries.

enhanced queries are scored using training question answer pairs. Building a Query Content Model does not replace traditional query expansion - both processes can be applied simultaneously to the test questions: specific keywords and knowledge derived from new questions are the basis for traditional query expansion and the clustering of similar training questions is the basis for learning additional content conducive to better retrieval performance. Through the Query Content Model we allow shared context to play a more significant role in query generation.

Some QA systems already shape queries differently according to their type and enhance them with additional content. For example, for the "Who is X?" type of questions, words such as 'biography' and 'profession' can also be included in the query.. However, this process is usually rudimentary and is performed manually when answering strategies are implemented, associating question types with specific additional keywords. In our instance-based QA approach, query content is learned and does not require expert human knowledge in writing and selecting content features.

The Query Content Model takes advantage of cluster-specific data and learns from training questions how to build better queries from a content perspective. This approach can also be extended in order to learn the query structures that are best suited for retrieving relevant documents for questions in a specific cluster.

7.3.2 Scoring Enhanced Queries

The field of information retrieval several precision and recall based metrics that can be applied to measuring retrieval performance in QA. Precision $P = r/n$ is the ratio of the number of relevant documents retrieved r to the number of documents retrieved n, and recall $R = r/q$ is the ratio of the number of relevant documents retrieved r to the number of relevant documents in the corpus R.

The *number of relevant documents retrieved* r is a very simple statistic that does not take into account document ranking. It has the advantage of directly specifying the upper bound for the answer extraction component.

R-precision r is the precision at the rank of the last of a pre-specified k number relevant documents. This statistic can be a good indicator of density, especially if the answer extraction performs well when it observes a minimum number of relevant documents. R-precision, however, is not sensitive to ranking: it does not differentiate between different distributions of the top $k-1$ relevant documents. Moreover, a pre-specified number of relevant documents has to be selected in order to compute R-precision. Depending on the question type, topic density in the local corpus or on the web, it is difficult to set a specific recall threshold.

Average precision P_A is a measure of retrieval performance that takes into account ranking as well as answer density:

$$P_A = \frac{\sum_{i=0}^{r} P(\text{rank of } rd_i)}{r} \tag{7.2}$$

where rd_i is the i_{th} relevant document retrieved, and $P(rank)$ is the precision computed on the set $\{0, rank\}$ of retrieved documents.

Although it is not easily interpretable, average precision has the advantage of sensitivity to the overall ranking, stability to small changes in ranking, and contains both precision and recall factors.

Question answering systems typically observe a strong dependency between the retrieval component and the answer extraction component. Sub-optimal retrieval may perform well if answers **can** be extracted from the available text. On the other hand, sub-optimal extraction may perform well given documents with simple appropriate context. During training, an indirect measure of document retrieval performance can be propagated back from the answer extraction step. The extraction step's performance can be evaluated using a weighted harmonic mean between precision and recall [120], called *F-measure*:

$$\text{F-measure} = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)RP}{\beta^2 P + R} \tag{7.3}$$

where β is a parameter signifying the relative importance of precision P and recall R. The most commonly used is balanced F-measure – i.e. $\alpha = 1/2$ and $\beta = 1$.

A query's score – the probability of success – is given by the number of correct answers extracted as well as by the extraction precision. In effect this backward propagation reflects allows us to realistically estimate the probability of success at every step in a linear strategy – one cluster, one query type, one extraction model.

This measure is flexible and is answer extraction dependent – i.e. it factors in how good the answer extraction model of a QA particular system is. This allows the QA system to exploit the symbiotic relationship between retrieval and extraction in the QA process and evaluate their performance at the simultaneously.

7.4 Retrieval Experiments and Results

We tested the performance of cluster-based enhanced queries and compared it to the performance of simple keyword-based queries, and queries expanded through synonyms and inflectional variants. We also compare several feature selection methods used to identify content features that are conducive to successful cluster-based queries.

To acquire sufficient data for a thorough experiment with multiple retrieval strategies, the instance-based system used the Google API ($www.google.com/apis/$) for document retrieval. The documents are filtered for known question answering content (documents including terms such as “trec”, “aquaint”, “nist”, ’question answering”, TREC document ids, or the question itself), the html tags are removed, and the emerging text documents are divided into sentences. The number of documents retrieved and processed specifically for all the simple and expanded queries is approximately $300,000$ and total size of the retrieved dataset is approximately $10GB$.

For each new question, we identify the training questions that share a minimum surface structure (in-order set of words) which we consider the prototype of a cluster. This constraint-based approach has the advantage of generating clusters of different granularity and different number of training instances for the same question. Each cluster represents a different, implicit notion of question similarity based on the set of training questions it covers. Therefore different clusters lead to different retrieval strategies and different answer extractors. These retrieval experiments are restricted to using only clusters of size four or higher to ensure sufficient training data for learning queries from individual clusters.

In the context of question answering, for any specific question the set of relevant documents in a local corpus is usually unknown. Moreover, document and passage relevance is judged according to a set of answer keys in the form of regular expressions. For most questions, these regular expressions are incomplete, they don’t cover all possible correct answers, nor do they cover all surface forms of the same answer. For example “*Mr. Rogers’ show*”, and “*Mr. Rogers’ Neighborhood*” can both be correct answers to the same question.

Since we are interested in obtaining accurate retrieval performance measurements for question answering, we attempted to avoid most of the above pitfalls in our retrieval-focused experiments by performing experiments using all temporal questions from the TREC 8-12 evaluations. Temporal questions have the advantage of having a more restrictive set of possible answer surface forms, which lead to a more accurate measure of retrieval performance. At the same time temporal questions cover both more difficult questions such as “*When was General Manuel Noriega ousted as the leader of Panama and turned over to U.S. authorities?*” as well as simpler questions such as “*What year did Montana become a state?*”. We employed this dataset for a more in-depth analysis of IBQA retrieval performance and strategies. However, for the overall instance-based question answering experiments, we employ the entire TREC collection.

Four sets of queries are generated and their performance tested. We are interested in observing to what extent additional methods produce additional relevant documents. The initial set of queries are constructed by simply using a bag-of-words approach on the question keywords. These queries are run through the retrieval engine, each generating 100 documents. The second set of queries builds on the first set, expanding them using synonyms. Each word and potential phrase is expanded using synonyms extracted from WordNet synsets. For each enhanced query generated, 100 documents are retrieved. To construct the third set of queries, we expand the queries in the first two sets using inflectional variants of all the content words (e.g. verb conjugations and noun pluralization). For each of these queries we also retrieve 100 documents. All experiments were performed using leave-one-out cross validation.

When text corpora are indexed without using stemming, simple queries are expanded to include morphological variations of keywords to improve retrieval and extraction performance. Inflectional variants include different pluralizations for nouns (e.g. *report*, *reports*) and different conjugations for verbs (e.g. *imagine, imagines, imagined, imagining*). Under local corpus retrieval inflectional expansion bypasses the unrelated term conflation problem that stemmers tend to have, but at the same time, recall might be lowered if not all related words with the same root are considered. For a web-based question answering system, the type of retrieval depends on the search-engine assumptions, permissible query structure, query size limitation, and search engine bandwidth (allowable volume of queries per time). By using inflectional expansion with queries that target web search engines, the redundancy for supporting different word variants is higher, and has the potential to increase answer extraction performance.

For the fourth and final set we employ our cluster-based query expansion method. These queries incorporate ngrams and paraphrases learned from the training questions covered by the same cluster. Instead of further building an expansion using the original question keywords, we expand using contextual features that co-occur with answers in free text. For all the training questions in a cluster, we gather statistics about the co-occurrence of answers and potentially beneficial features. These statistics are then used to select the best features and

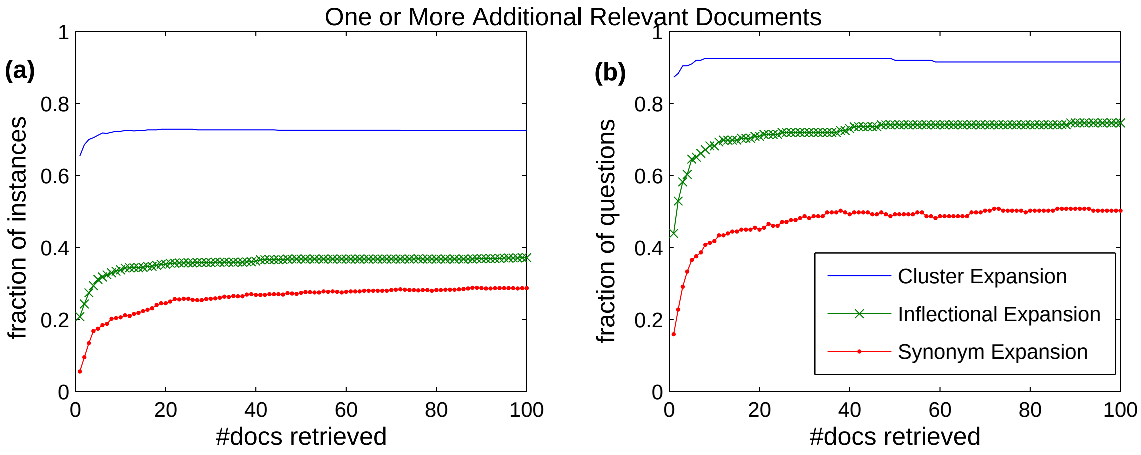

Figure 7.4: The benefit of iteratively adding each expansion method: (a) the fraction of cluster instances and (b) the fractions of questions (aggregated across clusters) that have at least a new (different) relevant document in addition to previous methods.

apply them to new questions whose answers are unknown. Figure 7.4(b) shows that approximately 90% of the questions **consistently** benefit from cluster-based query expansion when compared to approximately 75% of the questions when employing the other methods combined. Each question can be found in multiple clusters of different resolution. Since different clusters may lead to different selected features, questions benefit from multiple strategies and even though one cluster-specific strategy cannot produce relevant documents, other cluster-specific strategies may be able to. When aggregating results from individual clusters, we are only concerned about cluster-specific question instances (a). When aggregating results from individual questions (b), multiple clusters contribute with different features (strategies) and benefit retrieval performance.

The cluster-based expansion method can generate a large number of contextual features. When comparing feature selection methods, we only select the top 10 features from each method and use them to enhance existing question-based queries. Furthermore, in order to retrieve, process, extract, and score a manageable number of documents, we limited the retrieval to 10 documents for each query. In figure 7.4 we observe that even as the other methods retrieve more documents, $\sim\ 90\%$ of the questions still benefit from the cluster-based method. In other words, the cluster-based method generates queries using a different type of content and in turn, these queries retrieve a different set documents than the other methods. This observation is true even if we continue to retrieve up to 100 documents for simple queries, synonym-expanded queries, and inflexional variants-expanded queries.

This result is very encouraging since it suggests that the answer extraction components of question answering systems are exposed to a different type of relevant documents, previously inaccessible to them. Through these new relevant documents, cluster-based query expansion provides extractors with richer and more varied sources of correct answers for 90% of the questions. While there is a strong performance correlation between retrieval and extraction in the context of question answering, better retrieval performance does not guarantee better overall question answering performance – different answer extractors may or may not take advantage of additional relevant documents. However, if retrieval performance is poor, answer extraction performance is also likely to be poor. Moreover, if the answer extractors are provided with more of the same documents from which they couldn't previously extract correct answers, relevant document density is not a good indicator of retrieval performance. Previous work focused mostly on different ways of measuring retrieval performance without considering the tightly coupled answer extraction requirements. Our experiments show that cluster-based expansion addresses this issue and supplies answer extractors with a set of documents that cover a different part of the relevant document space.

	new relevant documents	
simple	4.43	100%
synonyms	1.48	33.4%
inflect	2.37	53.43%
cluster	1.05	23.65%
all	9.33	210.45%
all - synonyms	7.88	177.69%
all - inflect	6.99	157.69%
all - cluster	8.28	186.80%

Table 7.3: Keyword based queries ('simple') and expansion methods based on synonyms, inflectional variants, and cluster-based. Shows the average number of additional relevant documents across instances at twenty documents retrieved.

Although expansion methods generate additional relevant documents that simpler methods cannot obtain, an important metric to consider is the density of these new relevant documents. We are interested in the number/percentage of new relevant documents that expansion methods contribute with. Table 7.3 shows at retrieval level of twenty documents how different query generation methods perform. We consider keyword based methods to be the baseline and add synonym expanded queries ('synonym'), inflectional variants expanded queries ('inflect') which build upon the previous two types of queries, and finally the cluster enhanced queries ('cluster') which contain features learned from training data. We see that inflectional variants have the most impact on the number of new documents added, although synonym expansion and cluster-based expansion also contribute significantly.

7.4.1 Feature Selection for Cluster-Based Retrieval

Content features are learned from the training data based on observing their co-occurrences with correct answers. In order to find the most appropriate content features to enhance our cluster-specific queries, we have experimented with several feature selection methods [134]: information gain, chi-square, the phi coefficient, and simple conditional probability as a baseline.

Information gain (IG) in the context of our question clustering problem measures the reduction in entropy for the presence/absence of an answer in relevant passages, when we know whether an n-gram feature is present in these passages or not:

$$IG(f,a) = \sum_{b_f \in \{f, \neg f\}} \sum_{b_a \in \{a, \neg a\}} P(b_a, b_f) log \left(\frac{P(b_a, b_f)}{P(b_a)P(b_f)} \right)$$

where b_a and b_f represent the presence or absence of an n-gram and of an answer in a passage. We combine the information gain of each n-gram at the cluster level by averaging over individual questions in the cluster:

$$IG_C(f) = \sum_{q \in C} P(q|C) IG(f,a)$$

Chi-square (χ^2) is a non-parametric measure of association that quantifies the lack of independence between two variables - in our case the passage-level association between an n-gram feature f and a correct answer a. Since we are interested in evaluating the usefulness of each feature with respect to each cluster C, we combine the question-level χ^2 statistics:

$$\chi^2_C(f) = \sum_{q \in C} P(q|C)\chi^2(f,a)$$

where $P(q|C)$ is the probability that the question q belongs to cluster C. χ^2 takes values ranging between zero and infinitely large positive numbers.

When applied to question clustering, chi-square tests whether there is a significant difference between the distribution of n-gram feature f and the distribution of correct answers a in the passages. However, judging the relative usefulness among features using the $[0,\infty)$-valued χ^2 statistic is difficult.

Phi (ϕ) is a transformation that compresses the values of chi-square into the $[0,1]$ interval, allowing us to measure for individual questions the degree of association between an n-gram feature f and the presence/absence of an answer a in relevant passages:

$$\phi(f,a) = \sqrt{\chi^2(f,a)/N}$$

phi is often interpreted as a Pearson correlation coefficient. Similar to the χ^2 case, we combine the evidence for feature f from all questions in each cluster:

$$\phi_C(f) = \sum_{q \in C} P(q|C)\phi^2(f,a)$$

As a baseline, we also considered whether the presence of feature f is a good predictor of the presence of answer a in passages:

$$D_C(f) = \sum_{q \in C} P(q|C)P(a|f)$$

In figure 7.5 we compare these feature selection methods on our dataset. The selected features are used to enhance queries and retrieve additional documents. We measure the fraction of question instances for which enhanced queries obtain at least one new relevant document. The comparison is made with the document set generated by keyword based queries, synonym expansion, and inflectional variant expansion.

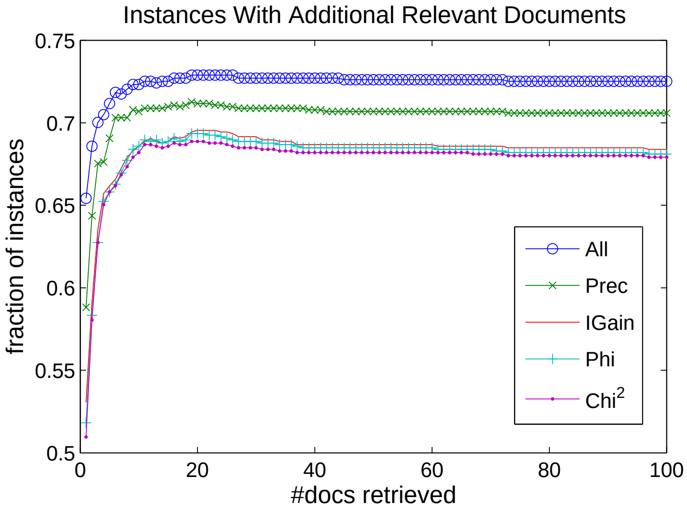

Figure 7.5: Feature selection methods - corresponding enhanced queries performance on test data. Retrieval performance is measured here as the fraction of question instances that benefit with at least a new relevant document from enhanced queries. 'All' represents combining the features from all feature selection methods.

In this experiment, average precision on training data is the best predictor of additional relevant documents: approximately 71% of the test question instances benefit from queries based on average precision feature selection. However, the other feature selection methods also obtain a high performance: approximately 68% of the test question instances benefit from these methods.

Since these feature selection methods are different in nature, it is interesting to see the performance of their combination ('All') and as expected, we observe (figure 7.5) a performance boost from feature set merging (73%). In this case there is a trade-off between a 2% boost in performance and an almost double set of features and enhanced queries. This translates into more queries and more documents to be processed. Although it is not the focus of this research, we note that a clever implementation of IBQA might incrementally add features from the next best selection method only after the existing queries and documents have been processed. This approach lends itself to be a good basis for utility-based models and planning [93, 48].

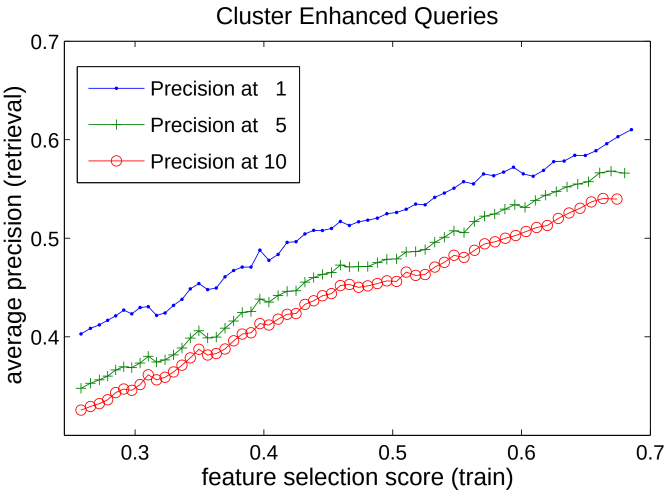

Figure 7.6: The average precision of cluster enhanced queries correlates well with the scores generated by feature selection based on the training data.

An important issue in these experiments is to what extent the scores of the selected features are meaningful and correlate with actual retrieval performance on test data. We measure the average precision of these queries at different number of documents retrieved. Figure 7.6 shows precision at one, five, and ten documents retrieved. An initial observation is that feature scores do indeed correlate well with actual retrieval performance. This result is confirmed by all three curves and suggests that useful features are in fact learned. Another important observation is that average precision when one document is retrieved is consistently greater than precision at five documents retrieved, which in turn is greater than precision at ten documents retrieved. This result shows that the rank of a document correlates to the relevance of that document. In other words, the higher the rank of the document, the more likely it is to be relevant, which is a desirable quality in information retrieval.

Furthermore, we are interested to see how individual feature selection methods vary with the number of documents retrieved - whether rank and average precision are corelated. Figure 7.7 shows that selection based on training data average precision and Chi^2 yields the

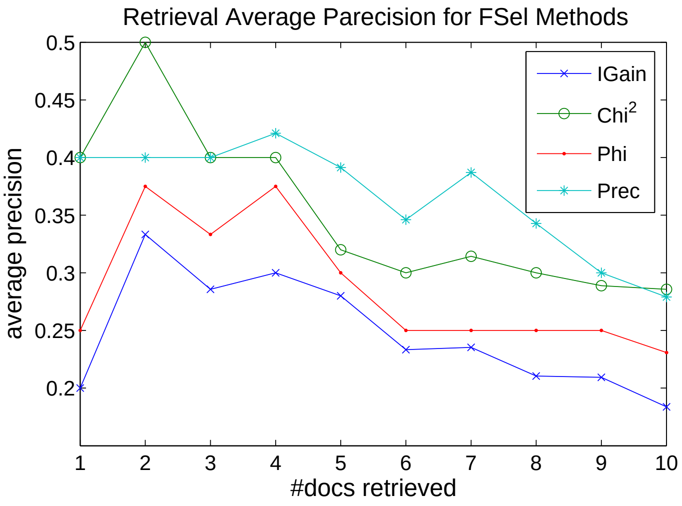

Figure 7.7: Performance of different feature selection methods as a function of the number of documents retrieved.

best performance on test data in terms of average precision. The common tendency among all feature selection methods is to have better performance with higher document ranks.

Some cluster enhanced queries are very specific and focused on retrieving exact contexts for correct answer: “he died on ANSWER of heart failure”, “and ANSWER was the year of her birth”, “a native of ANSWER , Mr.” etc. Because of being so specific, most of these queries are able to retrieve a limited number of relevant documents with high rank and high precision. However, as we retrieve more documents, it is less likely to find the same features present in these documents. Another category of cluster enhanced queries is based on more generic features that guide retrieval with lower precision, but higher recall: “born in”, “native of”, “the first to” etc. By generating both types of queries, the Query Content Model is able to find new relevant documents that are usually not found by more traditional methods.

7.4.2 Qualitative Results

During the relevance feedback process based on individual clusters, several artifacts came to light. For several of the clusters, we observed that the feature selection process, consistently and with high confidence selected features such as “*noun NP1 has one meaning*” where $NP1$

is the first noun phrase in the question – and is different for different question instances in the cluster.

The indirect reason for selecting such features is in fact the discovery of authorities: websites that follow a particular format and which have a particular type of information, relevant to a cluster. In the example above, the websites *answers.com* and *wordnet.princeton.edu* consistently included answers to clusters relevant to a person's biography. Similarly, *wikipedia.org* often provides answers to definitional questions (e.g. "*what is uzo?*").

Question: When did Bob Marley die?

The noun Bob Marley has one meaning:

Meaning #1: Jamaican singer who popularized reggae (1945-1981)

** Born:*	*6 February 1945*
** Birthplace:*	*St. Ann's Parish, Jamaica*
** Died:*	*11 May 1981 (cancer)*
** Best Known As:*	*The reggae hero who did "Get Up, Stand Up"*

In the example above, profiles for many entities mentioned in a question cluster were found on several *authority* websites. For the entity "Bob Marley", the answer to the year of death question can easily be found. In fact, this observation has the potential to lead to a cluster-based authority discovery method, in which certain sources are given more credibility and are used more frequently than others. For example, by observing that for most questions in a cluster, the *wikipedia* site covers at least one correct answer (ideally that can actually be extracted), then it should be considered (accessed) for test questions before other sources of documents. Through this process, given a set of questions processed using the IBQA approach, a set of authority answer sources can be identified.

7.4.3 Selection for Document Retrieval

We assume a document to be relevant in the context of question answering if it contains a correct answer in a correct context. Since it is very difficult to automatically evaluate the correctness of context, the notion of relevance is sometimes relaxed to whether a document contains the correct answer, regardless of context. As shown in the previous sections, through cluster-specific data, the retrieval component of an instance-based question answering system system learns n-grams and features that improve retrieval when added to queries. The improvement is measured when these queries are used to retrieve documents for the questions in the same cluster. The learned features become part of the cluster-based answering strategy which can be applied to new similar questions.

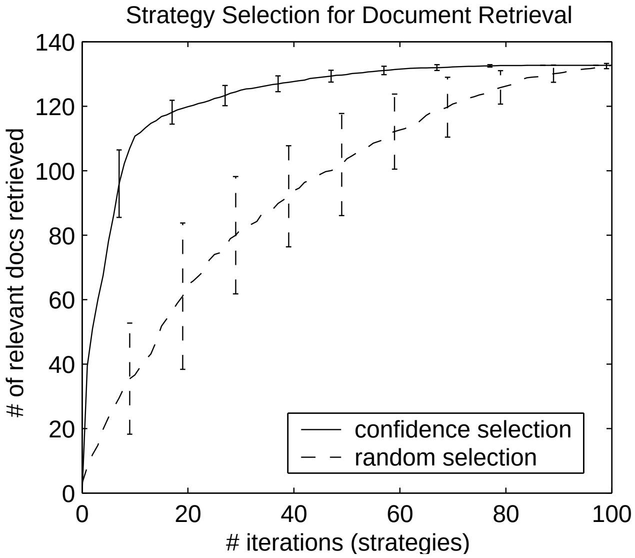

Figure 7.8: Smart selection based on strategy confidence allows the instance-based question answering system to employ only 10% of its available cluster-specific strategies to retrieve 80% of the accessible relevant documents.

When trying to answer the question "*When did Mozart die?*" it may be beneficial to create queries that contain features such as "*biography*", "*cemetery*", "*spent his life*", "*sacrificed himself*", etc. In many question answering systems the retrieval component contains rules for building better queries for specific types of questions – in our example: *time_of_death*. Under the cluster-based approach, these features are learned from other similar questions in the training data, and are then added to cluster-specific answering strategies. We measure the retrieval confidence $conf(A_{IR}(C_j)|q)$ of an answering strategy A derived from cluster C_j given a new test question q:

$$conf(A_{IR}(C_j)|q) = P(d^+|A_{IR}(C_j)) \cdot P(C_j|q) \cdot s(j) \tag{7.4}$$

where $P(d+|A_{IR}(C_j))$ is the probability of retrieving a relevant document d^+ using strategy $A_{IR}(C_j)$ and is measured by testing its effectiveness on a held-out set of questions from the cluster. $P(C_j|q)$ is the probability of a cluster containing questions similar to q and is given by the average similarity between q and q_j ($i \in C_j$) normalized over all clusters. $s(j)$ is a minimal cardinality condition for clusters.

Figure 7.8 shows the effect of using confidence-based selection in order to iteratively add appropriate answering strategies (i.e. beneficial query content). The more strategies are employed to create queries and retrieve new documents, the less time will be available for answer extraction and answer merging. The iterative process offers a good trade-off between performance and number of strategies used, as well as a good basis for user-defined utility functions. In our experiments, if the QA system selects only 10% of the available strategies, the retrieval performance is approximately 80% of the maximum achievable using the existing current strategies.

7.5 Query Content Modeling – Summary

We introduce a new, cluster-based query expansion method that learns queries which are successful on multiple similar questions. Since it is orthogonal to traditional query expansion methods, it can be used in addition to pseudo-relevance feedback. Traditional QA query expansion uses only the individual keywords in the test question and expands them using various semantic and corpus-based methods. In contrast, the cluster-based expansion learns features from context shared by similar training questions from that cluster.

Since the features of cluster-based expansion are different from the features used in traditional query expansion, the main benefit of the retrieval step in a QA system is that it brings in new relevant documents that are different from the ones retrieved using the existing expansion techniques. Our experiments show that more than 90% of the questions benefit from our cluster-based method when used in addition to traditional expansion methods.

In this chapter we have presented experiments with several feature selection methods and we have shown that using average precision as the selection method works best unless all selection methods are used in conjunction, in which case the number of queries generated might be prohibitive. Experiments also show strategy selection impact on our IBQA implementation: by selecting 30% of the strategies, we can obtain close to 90% of the full-strategy system performance.

CHAPTER 8

Answer Extraction

Contributions: *In answer extraction for QA, this thesis proposes extraction models are trained on abstracted, aggregated cluster-specific data. The training data originates from documents retrieved for multiple similar questions, leading to more focused answer extractor models.*

The answer extraction component of a question answering system is one of the most critical but also one of the most difficult stage in the process of finding exact correct answers to questions. Given a segment of text (e.g. document, passage, sentence), an answer extractor identifies candidate answers and makes a decision whether each candidate is a correct answer or not. Most question answering systems' answer extractors compute scores based on their content and structure, as well as on the content and structure of the corresponding textual contexts.

Text segments vary in length, depending on the specific implementation of the question answering system and are usually in the form of documents, passages, or individual sentences. The type of data varies depending on the particular corpus used (e.g. AQUAINT, the web) and also on the article genre (e.g. news story vs. stock market report). For web-based question answering, the variations are even greater and extraction modules have to be robust enough to be able to process news stories, biographies, as well as lower quality webpages such as product descriptions, fan sites, logs, and less coherent forums. Another drawback for web-based extraction is the lack of surface-level and content-level standardization. For example, while one author might be very careful about grammar and casing (lower case vs. upper case), another might be more informal and might be less inclined to pay attention to these issues. At the same time, source veridicity is often a problem when processing web content and very often less reliable sources could derail the QA process. However, this problem can often be solved by merging similar answers and attempting to find multiple sources that support the same conclusion, as a form of implicit verification.

In the best case scenario, text segments processed by answer extractors are relevant - i.e. contain a correct answer. However, in more realistic scenarios, this is very often not true. The pre-extraction retrieval stage may not be able to obtain relevant documents due to unavailability of relevant documents in the corpus or due to inadequate queries. Also, the varying proportion of correct candidate answers and incorrect candidate answers (positive and negative examples) varies among question types and in the case of instance-based QA among clusters. This makes an even more critical and compelling reason to train individual extractors for individual clusters.

For the TREC-style questions, text segments that have self-contained answers for the QA task, typically range between one and three sentences in size. Very often, the answer can be correctly identified from only one sentence. However, nominal and pronominal references sometimes occur across sentence boundaries and make extraction more difficult. Answer extraction is more precision-oriented component in the QA process since it is more important to obtain a correct answer with high confidence than to obtain several correct answers with similar confidence to incorrect answers. Because of this, given sufficient relevant documents, answer extractors often focus on obtaining confident correct answers from very clear context (e.g. “*Mozart died on Dec 5, 1791*”) at the cost of missing correct answers in more ambiguous contexts (e.g. *"Mozart ... treated by Dr. Fitzgerald ... praises from Franz Schubert. A year later he did on Dec 5th."*).

Depending on the task, an answer extractor may identify very small, factoid candidates which are well defined text snippets [124], paragraphs [110], or a collection of text segments containing *nuggets* [125] that are relevant in answering questions. For factoid TREC-style questions, very often well-defined snippets are most appropriate, even though the set of surface forms of correct answers is often large. For definitional questions phrases are very often sufficiently complex and answer the original questions. However, for completeness, several methods [125, 66] have been developed to deal with relevant text nuggets that might appear in a long answer or several answer components. For FAQ-type questions, how-to, and why-type questions, longer paragraphs and sometimes multi-document paragraphs constitute the answers. For these types of questions machine translation-inspired metrics are more appropriate [110]. However, since these types of questions are not the focus of this work, we will not explore answer extractors that cover them.

The performance of an answer extraction component is intertwined with the performance of the retrieval component of a QA system. If the retrieved documents are not relevant - i.e. do not contain correct answers -, the answer extractor becomes inconsequential since the overall performance will certainly be low. However, if the retrieved documents are all relevant, but the structure of the text is too complex, the correct answers also cannot be extracted and the performance of the retrieval component is irrelevant. Hence, the goal is to find a retrieval-extraction strategy that yields the best performance for a particular QA system.

8.1 Related Work

The task of named entity tagging is very much related to answer extraction. BBN's Nymble system [8] and subsequently IdentiFinder system [9] successfully employed a Hidden Markov Model (HMM) approach to the task of recognizing and classifying named entities such as names, locations, organizations. The results obtained on English text were consistently above

0.9 F-measure and similar results in non-English text. In the context of automatic content extraction (ACE) the extraction task consists of first identifying mentions of entities such as named entities (e.g. names, locations, organizations), pronominal entities (e.g. he, hers), and nominal entities (e.g. the president, the salesman's daughter). After these mentions of entities are identified, reference resolution is performed (e.g. mentions such as *he*, *Clinton*, and *the president* could be one and the same person). Finally a more difficult task under ACE is classifying the role and relationships among entities. The maximum entropy IBM system [34, 73] consistently performed well in identifying events and relations from text from multiple sources, in multiple languages and forms.

Initial answer extraction experiments focused on answer types that corresponded to named entities. AT&T's QA system [2] performed entity extraction using the Cass partial parser based on finite-state cascades [1]. The SMU Falcon system [42] successfully employed more extensive answer types and explored dependencies between words in order to better capture the semantics of questions and answers. It also combined semantic and syntactic knowledge with empirical methods in order to obtain a good performance on TREC data.

Answer extraction can also be viewed as a classification problem in which answer correctness is evaluated based on various features derived from both the question and the candidate answer. The CMU Javelin system [93] trains support vector machines, k-nearest neighbor, and decision tree classifiers to evaluate the correctness of individual answers. Under this approach candidate answers are identified and then their correctness assessed by different classifiers for different answer types. A maximum-entropy approach employed by IBM [55] computes the probability of correctness given the question and the candidate answer by introducing a hidden variable which represents the answer type (i.e. named entity). The model uses sentence features, entity features, definition features, and linguistic features in order to capture various aspects of the question-answer relationship.

A noisy channel approach has been proposed by ISI [32] that maps sentences containing a possible answer into the original question. This approach attempts to measure the distance between sentences and questions. In this framework, the desired outcome is for short distances to imply answer correctness and long distances to imply incorrect answers. With the approach yielding moderately good results, it interesting to note that the ISI question answering system is based on publicly available software.

Another approach to answer extraction is to apply simple patterns based on surface forms or part-of speech tags in order to build an off-line database for specific question types – e.g. who-is type questions [33]. The answers are usually extracted from large corpora. While the approach has the benefit of acquiring off-line answers, specific extraction patterns have to be written for many different question types. Large collections of surface patterns have been employed successfully [111, 16, 45] in the TREC question answering task. Learning these surface patterns as regular expressions [105] can be achieved by using bootstrapping methods on web data and using suffix trees for determining optimal answer length.

Many QA systems exploit redundancy of candidate answers, by using answer frequency to boost the confidence of correct answers. Web documents tend to provide answer redundancy in simple contexts. This fact has been previously used to show that the most frequent answers are usually the correct ones [31]. Redundancy has also been used to cluster multiple answers with low confidence and generate a representative answer with higher confidence [16, 2, 21, 62].

Most systems also make use of specific resources such as dictionaries, encyclopedias, and gazetteers and online semi-structured sources [72]. Definition questions in particular can be answered by using various online and offline knowledge sources [47, 67]. BBN's question answering system performed [132] very well on definition questions by building a question/answer profile from online sources such as Merriam-Webster, Columbia Encyclopedia, Biography.com, and Wikipedia. Answers extracted from local corpora using patterns are then ranked against the profile.

8.2 Answer Extraction under IBQA

The answer extraction model (Figure 8.1) is the component in the answering strategy where actual answers are being extracted from the enhanced documents obtained in the retrieval

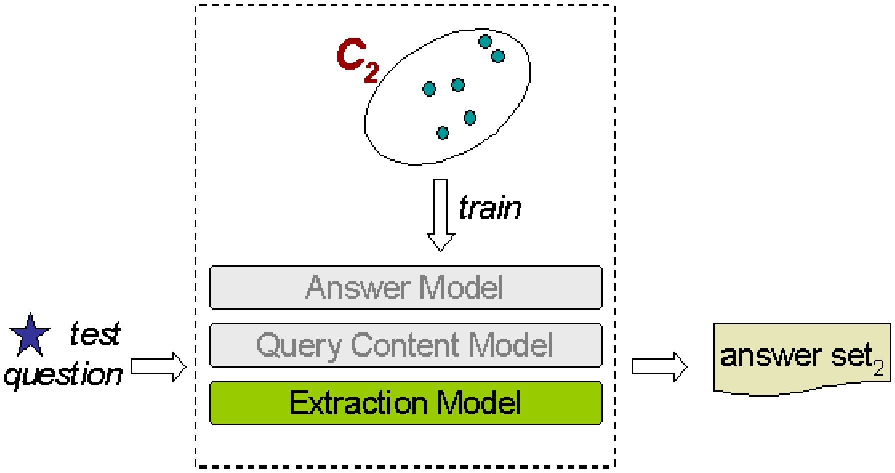

Figure 8.1: Answer extraction modeling as the third component of an answering strategy.

stage. Both rule-based answer extraction and statistical answer extraction make the implicit or explicit assumption that answers to similar questions usually have similar structure. Furthermore, answers usually occur in contexts with similar content and structure. Based on these assumptions, the extraction methods propose rules or statistical models that exploit structure and context for particular answer types and extract possible answers. Under the instance-based question answering approach, clusters provide sets of similar training questions. We use these sets of questions as raw training data to build cluster-specific extraction models. The extraction models become part of answering strategies for individual clusters. When clusters are deemed relevant to new test questions (i.e. questions in the cluster are also similar to the new question), their corresponding answering strategies are activated and the cluster-specific models are applied. When the questions are indeed similar, the cluster-based extractor is able to extract correct answers and the cluster strategy is successful. If on the other hand, the cluster questions are only similar according to dimensions that cannot be exploited by the answer extraction method, the cluster strategy will not be successful. However, since each question is covered by several clusters, several extractors trained on different question sets will be activated, increasing the likelihood that at least a strategy will be successful.

Consider the example in table 8.1, which shows four training questions corresponding to the cluster “When was <NP> <VB>?”. The table describes how to obtain a level of abstraction over the raw training data from multiple questions. This abstraction allows text from multiple questions to become more homogeneous and to be used for building a more focused extractor. Some of the relevant documents retrieved in the IR stage will contain both correct answers and incorrect answers to the questions. In order to present an answer extraction learner with homogeneous training data, the following steps are taken:

Algorithm 5: Data Abstraction Procedure

```
1: sentences that contain at least an expanded question keyword are selected from documents retrieved for training questions in the cluster
2: text fragments that match the expected answer type are identified
3: sentences with correct text fragments are marked as positive examples
4: sentences without correct text fragments are marked as negative examples
5: keywords and answers are abstracted from the sentences and replaced with placeholders – e.g. NP, VB, see table 8.1
6: extract features from abstracted data
7: train cluster-specific answer extractor on this data
8: use extractor on documents retrieved for the test question
```

One of the more successful approaches to answer extraction consists of answer correctness classification. Candidate answers are identified in text as possible answers to the question, then using context-based features, each candidate is classified into two classes: correct and incorrect. The classification score is usually used as the answer confidence. Similar to using an answer type hierarchy [93], the an IBQA system can use the expected answer type distribution and find segments of text which are possible answers. The expected answer type is thus similar to answer types observed in conjunction with other questions from the same cluster. These segments of text are considered candidate answers and are used as target data points to be labeled as correct or incorrect. The goal is to build a model that accepts snippets of text and decides whether they contain a correct answer or not.

After abstracting away the question-specific keywords from relevant documents, we identify possible candidate answers and extract from their contexts a set of feature vectors. The

Cluster:	When was <**NP**> <**VB**>?
Instances:	When was <**the White House**> <**built**>?
	When was <**the first stamp**> <**issued**?
	When was <**the city of New Orleans**> <**founded**>?
	When was <**the telegraph**> <**invented**>?
	…
Contexts:	…**the White House** was **built** by an Act of Congress in 1790.
	…in May 6, 1840, **the first stamp** was **issued** in Great Britain …
	In 1718 …**founded the City of New Orleans** and named it.
	…Samuel Morse **invented the telegraph** in 1837, a machine …
	…
Training contexts:	…**NP** was **VB** by an Act of Congress in **ANSWER**.
	…in **ANSWER**, **NP** was **VB** in Great Britain …
	In **ANSWER** …**VB NP** and named it.
	…Samuel Morse **VB NP** in **ANSWER**, a machine …
	…
Training answer types:	1790 - *date::year*
	May 6, 1840 - *date::full_date*
	1718 - *date::year*
	1837 - *date::year*
	…

Table 8.1: An answer extractor is trained on data aggregated from multiple questinos from the same cluster. We show the result of each data processing step needed in order to prepare and aggregate the data. The resulting contexts is used to extract features on which the extractor is trained. Known answers to training question instances and their corresponding contexts are presented as positive examples to the answer extractor. Relevant keywords are obscured in the training data in order to allow the answer extractor to more easily extrapolate to new questions.

purpose of feature vectors is to succinctly describe each candidate answer and its context using features such as the number of words and verbs, presence of keyword (placeholders) in the passage and their syntactic relationship to the candidate answer, presence of specific verbs or structures, n-grams, etc. Depending on the resources and processing tools available for the domain and language being considered we extract lexical, syntactic, and semantic local and sentence level features.

Using the feature vectors a cluster-specific classifier is trained (e.g. support vector machines) to evaluate each candidate answer and discriminate between two classes: correct and incorrect. When new question instances arrive, the already trained cluster-specific models are applied to the new, relevant text snippets in order to evaluate their correctness. The resulting classifier scores are used as answer extraction confidence scores.

Regardless of the method used for extraction, specific models are learned by training models on passages of text retrieved for all questions in individual clusters. The *correctness* of an answer identified by the model is the actual precision obtained on the training data. Correctness is different from the *quality* (e.g. the *F1* function) of the model, which usually reflects both precision and recall. However, when computing the overall probability that a candidate answer is correct, recall of the answer extractor has a smaller impact on QA performance.

8.2.1 Feature Extraction

The IBQA extractors implemented in this work are trained on features extracted from cluster-specific data. As seen in the previous chapter, a retrieval model is learned for each individual cluster. We use this model to run queries and retrieve documents for training questions in the same cluster. These documents are pooled at the cluster level from multiple similar questions and individual question terms appearing in raw text are obscured, becoming a unified corpus for training an extraction model. From this corpus we extract several statistics and use them to guide feature extraction.

In a preprocessing step, the question terms are matched in the cluster-specific training corpus. The keyword set is expanded using morphological variants as well as semantic expansion through WordNet [82]. We find synonyms for each question term and use the known WordNet synset frequency as weights in our expansion. Furthermore, a second stage of inflectional expansion is performed on the synonyms in order to capture their variations in text: i.e. conjugations (am, are, is), plural/singular(bottle, bottles;) etc. In order to make the data processing more concrete, consider the following example, in which we show how to perform data abstraction:

question:	When did Bob Marley die?
expanded:	[1] Bob Marley, [0.9] Robert Nesta Marley, [0.7] Marley, [0.2] Bob
answer:	May 11 , 1981
sentence:	On May 11 , 1981 Robert Nesta Marley died at the age of 36 .
abstract:	On **ANSWER QTERM** died at the age of 36 .

Note, that besides the expanded question terms, the answer *May 11, 1981* is also abstracted away. Through this process, supporting sentences for different questions in the cluster can be used jointly to train an extraction model.

Proximity features – we have implemented several features that describe how close the candidate answer is to known question terms and their expanded forms. The proximity functions used are

1. inverse linear $\alpha/dist$
2. inverse quadratic $\alpha/dist^2$
3. exponential $exp(-\alpha{\cdot}dist)$
4. linear discounted $1 - \alpha{\cdot}dist/100$

where α is a function-specific parameter to be tuned or learned. Instead of pre-specifying a proximity function to be used for specific questions, we allow the extraction model to use them collectively as proximity features. In addition, we include a sentence length normalized version of these features. The normalization provides relative information as to how far within the sentence answer candidates are found. All the proximity features are used in conjunction with individual question term t_i semantic expansion weights:

$$prox(a^*, t_i') = f_d(a^*, t_i') * w(t_i') \tag{8.1}$$

where a^* is the candidate answer, t_i is the question term variant found in the text, f_d is the distance function used for the proximity feature, and $w(t_i')$ is the semantic expansion weight of the question term.

Sentence statistics – based on the observation that sometimes certain answer types occur in much simpler contexts under certain conditions: e.g. in shorter sentences, answer is towards the end of the sentence, all but one of the question terms tend to be present etc. These statistics change from cluster to cluster, depending on the types of questions, the types of answers, and the source of the documents. Among the sentence statics that we use as features are: sentence length, location of the answer within the sentence, density of answer candidates in a sentence, and absence or presence of particular question terms.

Unigram features – very often, the tokens (words) in the sentence are a good joint indicator of the presence or absence of a correct answer. We build the cluster-specific vocabulary, thresholded by frequency (i.e. only tokens with frequency above a certain threshold are used) and we use individual word presence as features. To reduce the noise level and to potentially improve performance we employ feature selection (information gain or chi^2).

Pattern features – n-grams and paraphrases are very powerful features given that sufficient data is available. We collect n-grams (from 1-grams up to 9-grams) but restrict them to include either a question term or an answer: "*and QTERM was born*", "*on ANSWER , the*". This restriction is used to maintain the feature space relatively small and limit the processing of training data. Similar to unigram features, we use the presence and frequency of these patterns as features to for answer extraction.

8.2.2 Extraction Methods

The instance-based question answering framework is open and compatible to many existing extractors described in QA literature. It can accommodate manually built extractors as well as statistical methods. At the extraction stage in the question answering pipeline we have a corpus of documents retrieved using learned queries, the analyzed question and the expanded keywords, as well as a specific cluster of similar training questions. From this data, many types of features – some of which are shown above – can be extracted from the raw text. Once the question terms and the answers appearing in text are abstracted away, the corpus becomes consistent across questions such that training an extractor becomes feasible.

We have implemented three main extractors attempting to cover several types of extraction methods: a simple proximity extractor, a pattern-based extractor, and a support vector machine extractor. One of the main advantages behind these extractors is that if used simultaneously, different clusters will resonate better with different answer extraction methods, improving performance. For example, a pattern-based extractor could work very well for questions of the type "*When did* **NP VB***?*" (e.g. "*When did Mozart die?*", "*When did P&G open?*") while a more simple approach based on proximity would be more appropriate for larger clusters with more varied training questions such as "*When did* **QTERM+***?*" (e.g. "*When did Mozart die?*", "*When did Armstrong first walk on the moon?*").

The **proximity extractor** is based on the proximity features described above. The premise for this simple extractor is based on the observation that answers tend to occur in contexts that include original question terms. The closer the candidate answer is to the question terms and their weighted expansions, the more likely it is for the candidate answer to be correct. We employ two simple methods for integrating these features: a manual model over these features, or employing a statistical method such as regression or building a classifier. In the simplest proximity extraction approach, for each term t_i, we first we identify its expansion that has the highest proximity score:

$$t_i^* \leftarrow \operatorname*{argmax}_{t_i' \in var(t_i)} \sum_{t_i' \in var(t_i)} prox(a^*, t_i')) \tag{8.2}$$

where $var(t_i)$ is the set of all expansions of term t_i. The highest score expansion t_i^* of a term t_i in test question q is further used to compute an overall proximity score $score(a^*, q)$ for a particular answer candidate a^*:

$$\begin{aligned} score(a^*, q) &= \sum_{t_i^* \in q} prox(a^*, t_i^*)) && (8.3) \\ &= \sum_{t_i^* \in q} f_d(a^*, t_i^*) * w(t_i^*) && (8.4) \end{aligned}$$

Table 8.2 shows proximity processing for our previous simple example question: "*When did Bob Marley die?*" in four different contexts:

Tables 8.3 and 8.4 describe the experiments with our proximity-based extractor using different proximity functions, described above. We compare instance-level and question level performance using the MRR and Top5 metrics

The *inverse linear* (InvLinear) proximity function performs better at instance level both in terms of MRR and Top5 scores, while the extractor using the *linear* function performs best at the question level. *Inverse linear* tends to do well on more instances of the same questions, thus boosting the instance-level scores but being redundant at the question level. The sentence *normalized linear* (LinearNorm) function performs lower than its un-normalized counterpart. This result supports the idea that for a correct answer to be scorred appropriately, the local context (around answer terms and question terms) has a bigger impact than the sentence length. Scaling proximity distances has an overall negative effect distances because it dilutes strong local contexts found in longer sentences.

The *linear* proximity function consistently performs well at question level and obtains

a) On Thursday , May 23 , 1981 , the Honorable Robert Nesta Marley was given an official funeral by the people of Jamaica .

$$\begin{aligned} prox(a,t_0) &= f_d(|\textit{, the Honorable}|)\cdot w(\textit{Robert Nesta Marley}) \\ prox(a,t_0) &= f_d(dist=3)\cdot 0.9 \\ prox_{exp}(a) &= 0.9e^{-3\alpha} \end{aligned}$$

b) Editorial Reviews Amazon.com The legend of Bob Marley (1945 - 1981) is well served by this comprehensive and clear – eyed look at the turbulent life and times of the reggae great .

$$\begin{aligned} prox(a,t_0) &= f_d(|\textit{(1945 –}|)\cdot w(\textit{Bob Marley}) \\ prox(a,t_0) &= f_d(dist=3)\cdot 1.0 \\ prox_{exp}(a) &= e^{-3\alpha} \end{aligned}$$

c) On May 11 , 1981 Robert Nesta Marley passed away at the age of 36 .

$$\begin{aligned} prox(a,t_0) &= f_d(|\epsilon|)\cdot w(\textit{Robert Nesta Marley}) \\ prox(a,t_0) &= f_d(dist=0)\cdot 0.9 \\ prox(a,t_1) &= f_d(|t_0 = \textit{Robert Nesta Marley}|)\cdot w(\textit{passed away}) \\ prox(a,t_1) &= f_d(dist=1)\cdot 0.5 \\ prox_{exp}(a) &= 0.9 + 0.5e^{-\alpha} \end{aligned}$$

d) Bob 's death of cancer , at the early age 36 in 1981 , was shrouded in mystery .

$$\begin{aligned} prox(a,t_0) &= f_d(|\textit{'s } t_1 \textit{ of cancer , at the early age 36 in}|)\cdot w(\textit{Bob}) \\ prox(a,t_0) &= f_d(dist=11)\cdot 0.2 \\ prox(a,t_1) &= f_d(|\textit{of cancer , at the early age 36 in}|)\cdot w(\textit{death}) \\ prox(a,t_1) &= f_d(dist=9)\cdot 0.7 \\ prox_{exp}(a) &= 0.2e^{-11\alpha} + 0.7e^{-9\alpha} \end{aligned}$$

Table 8.2: Examples of contexts and proximity score computation for question "*When did Bob Marley die?*". α is a parameter that can be tuned or learned for different answer types: e.g. temporal, location.

	InvLinear	Linear	Exponential	LinearNorm
instance level	0.501	0.263	0.235	0.199
question level	0.445	0.478	0.431	0.362

Table 8.3: **Mean Reciprocal Rank (MRR)** for proximity-based extractors using different distance functions. We show MRR performance at the instance level and also at the question level, which is more relevant for the overall QA system performance.

	InvLinear	Linear	Exponential	LinearNorm
instance level	0.666	0.376	0.345	0.261
question level	0.581	0.641	0.621	0.490

Table 8.4: **Top5** for proximity-based extractors using different types of functions. We show how many instances and how many questions have at least a correct answer in the Top5 extracted answers.

a MRR of 0.501 and a Percentage Correct score (Top1) of 0.338. These results, however encouraging have been accomplished with a very simple method, albeit with careful processing, elaborate question term expansion, and by experimenting with several proximity functions. This method does not take advantage of surface form and structure of answer contexts and is most suitable for clusters of questions that are very difficult to extract surface form features (e.g. patterns) for. However, for clusters with high question similarity, more complex methods have the potential of performing better, taking advantage of more consistent shared answer contexts.

The **pattern-based extractor** is based on the assumption that simple statistics of the context around question terms and answers are sufficient when building generalizable model. Such models usually have high precision and low recall, which makes them ideal for highly focused clusters with a reasonable number of data points such as "*When did NP VB?*", but likely not very efficient when considering larger clusters such as "*When did QTERM+?*" which contains widely different types of questions. The pattern-based extractor acquires patterns in the form of n-grams around question terms and potential answers and collects frequency counts on how often these patterns co-occur with correct answers (positive) vs. incorrect answers (negative). The model considers only patterns whose positive frequency is above a certain threshold (e.g. 3) and appear in more than one question. There are two ways of computing the precision of a pattern: at a *micro* level, where the frequency counts for each question are aggregated across the cluster, and at a *macro* level, where the frequency counts are binary for individual questions.

The macro precision can be viewed as adhering to a multi-variate view in which a question is a binary vector over patterns, coding the presence of a pattern in at least one context for a particular question, regardless of how many times it appears.

$$b(pat, q^+) = \begin{cases} 1 & \text{if } \exists \text{ co-occurrence between pattern 'pat' and a correct answer (+)} \\ 0 & \text{otherwise} \end{cases} \tag{8.5}$$

We similarly define $b(pat, q^-)$ to be the co-occurrence of pattern 'pat' with incorrect candidate answers that match the answer type. The macro precision P_{macro} is defined as:

$$P_{macro}(pat, q) = \frac{b(pat, q^+)}{b(pat, q) + 1} \tag{8.6}$$

where $b(pat, q) = b(pat, q^+) + b(pat, q^-)$ and represents the presence of pattern 'pat' in any context of question q, regardless of answer correctness.

The micro precision follows a multinomial approach, under which captures pattern frequency information for the contexts (i.e. sentences) retrieved for one question.

$$B(pat, q^+) = \text{co-occurrence frequency between pattern 'pat' and a correct answer (+)} \tag{8.7}$$

We similarly define $B(pat, q^-)$ to be the co-occurrence frequency of pattern 'pat' with incorrect candidate answers that match the answer type. The macro precision P_{micro} is defined as:

$$P_{micro}(pat, q) = \frac{B(pat, q^+)}{B(pat, q) + 1} \tag{8.8}$$

where $B(pat, q) = B(pat, q^+) + B(pat, q^-)$ and represents the frequency of pattern 'pat' in any context of question q, regardless of answer correctness.

The micro precision can be highly unstable for example when a pattern co-occurs extremely frequently with correct answers for only one question in the cluster, but too specific. At the same time patterns that co-occur moderately often with correct answers for multiple questions, which are very useful, will have a lower precision. When using patterns that are highly discriminative of correct answers across questions will have a high macro precision, which is what we shall refer to simply as precision.

The pattern-based extractor scoring function can be computed in different ways that focus either on individual pattern precision or that combine precisions from several patterns. Under the simplest strategy, we select the pattern with highest precision and assign the corresponding candidate answer the same score: $score(a, q) = P_{macro}(pat^* a), q)$. Another strategy selects the best pattern associated with the candidate answer and also the best pattern associated with each question term and combines their scores:

$$score(a, q) = P_{macro}(pat^*(a), q) + \sum_{t_i \in q} P_{macro}(pat^*(t_i)) \cdot w(t_i) \tag{8.9}$$

Table 8.5 shows the pattern-based extractor processing for the simple example question: "*When did Bob Marley die?*" in the same four different contexts:

There are different types of patterns and pattern scoring methods that we explored in our instance-based QA experiments. Table 8.6 shows the MRR scores and table 8.7 shows the Top5 performance across instances and also across questions. Answer patterns are n-grams that include candidate answers – e.g. "*he was a great ANSWER ,*". We use the highest answer pattern precision (Ans) among patterns that include an answer candidate for scoring that answer candidate. We also use two types of candidate scoring through question term patterns. The first scoring involves finding the best question term pattern precision (QTerm) and using it to score the answer candidate. The second scoring method combines the best question term precisions from all the question terms in the original question (QTerm+) – e.g. by summing the precisions of the following patterns: "QTERM1 *'s cancer*" and "*he* QTERM2", where

a) On Thursday , May 23 , 1981 , the Honorable Robert Nesta Marley was given an official funeral by the people of Jamaica .

$$\begin{array}{lll}
pat^*(a) = \text{“\textbf{\textit{on ANSWER ,}}”} & b(pat^*(a), q^+) = 12; & b(pat^*(a), q^-) = 3 \\
pat^*(t_0) = \text{“}t_0 \text{ \textbf{\textit{was given an}}”} & b(pat^*(t_0), q^+) = 0; & b(pat^*(t_0), q^-) = 0 \\
\multicolumn{3}{l}{pscore_{simple}(a) = 12/(12+3+1) = 0.75} \\
\multicolumn{3}{l}{pscore_{combined}(a) = 0.75 + 0 \cdot 0.9 = 0.75}
\end{array}$$

b) Editorial Reviews Amazon.com The legend of Bob Marley (1945 - 1981) is well served by this comprehensive and clear – eyed look at the turbulent life and times of the reggae great .

$$\begin{array}{lll}
pat^*(a) = \text{“}t_0 \text{ \textbf{\textit{(}} } \bar{d}\bar{d}\bar{d}\bar{d} \text{ \textbf{\textit{- ANSWER)}}”} & b(pat^*(a), q^+) = 15; & b(pat^*(a), q^-) = 0 \\
pat^*(t_0) = \text{“}t_0 \text{ \textbf{\textit{(}}”} & b(pat^*(t_0), q^+) = 15; & b(pat^*(t_0), q^-) = 4 \\
\multicolumn{3}{l}{pscore_{simple}(a) = 15/(15+0+1) = 0.94} \\
\multicolumn{3}{l}{pscore_{combined}(a) = 0.94 + 0.75 \cdot 1.0 = 1.69}
\end{array}$$

c) On May 11 , 1981 Robert Nesta Marley passed away at the age of 36 .

$$\begin{array}{lll}
pat^*(a) = \text{“\textbf{\textit{on ANSWER}} } t_0\ t_1\text{”} & b(pat^*(a), q^+) = 7; & b(pat^*(a), q^-) = 0 \\
pat^*(t_0) = \text{“}t_0\ t_1\text{”} & b(pat^*(t_0), q^+) = 9; & b(pat^*(t_0), q^-) = 5 \\
pat^*(t_1) = \text{“}t_0\ t_1\text{”} & b(pat^*(t_1), q^+) = 9; & b(pat^*(t_1), q^-) = 5 \\
\multicolumn{3}{l}{pscore_{simple}(a) = 7/(7+0+1) = 0.875} \\
\multicolumn{3}{l}{pscore_{combined}(a) = 0.875 + 0.6 \cdot 0.9 + 0.6 \cdot 0.5 = 1.715}
\end{array}$$

d) Bob ’s death of cancer , at the early age 36 in 1981 , was shrouded in mystery .

$$\begin{array}{lll}
pat^*(a) = \text{“\textbf{\textit{age}} } \bar{d}\bar{d} \text{ \textbf{\textit{in ANSWER,}}”} & b(pat^*(a), q^+) = 5; & b(pat^*(a), q^-) = 1 \\
pat^*(t_0) = \text{“}t_0 \text{ \textbf{’s} } t_1\text{”} & b(pat^*(t_0), q^+) = 10; & b(pat^*(t_0), q^-) = 4 \\
pat^*(t_1) = \text{“}t_0 \text{ \textbf{’s} } t_1\text{”} & b(pat^*(t_1), q^+) = 10; & b(pat^*(t_1), q^-) = 4 \\
\multicolumn{3}{l}{pscore_{simple}(a) = 5/(5+1+1) = 0.714} \\
\multicolumn{3}{l}{pscore_{combined}(a) = 0.714 + 0.66 \cdot 0.2 + 0.66 \cdot 0.7 = 1.314}
\end{array}$$

Table 8.5: Examples of contexts and pattern score computation for question “*When did Bob Marley die?*”. Note that we allow answer patterns to include question terms, but we do not allow question term patterns to include answers, since the overall score would be based on redundant features. $pat^*(a)$ represents a pattern that includes an answer, whereas $pat^*(t_0)$ represents a pattern that includes only question terms, more specifically the term t_0. $b(pat, q^{+/-})$ is a frequency count of pattern co-occurrence with sentences with and without correct answers.

QTERM1 might be "*Bob Marley*" and QTERM2 might be "*passed away*". Finally, we also combine answer pattern precisions with question pattern precisions into a joint score. Scores are normalized and used as the final pattern-based extractor confidence.

	Ans	QTerm	QTerm$^+$	Ans & QTerm	Ans & QTerm$^+$
instance level	0.628	0.658	0.678	0.655	0.622
question level	0.869	0.880	0.882	0.864	0.858

Table 8.6: **Mean Reciprocal Rank (MRR)**[1] for pattern-based extractors using different types of pattens. We show MRR performance at the instance level and also at the question level, which is more relevant for the overall QA system performance.

	Ans	QTerm	QTerm$^+$	Ans & QTerm	Ans & QTerm$^+$
instance level	0.722	0.757	0.760	0.774	0.745
question level	0.909	0.914	0.914	0.904	0.909

Table 8.7: **Top5**[1] for pattern-based extractors using different types of pattens. We show how many instances and how many questions have at least a correct answer in the Top5 extracted answers.

From tables 8.6 and 8.7 we observe that the pattern based extractors that inclue question terms and answer terms have the highest performance: 0.914 Top5 score and 0.882 MRR score[1]. These scores are a considerable improvement over the proximity extractor performance since they capture the local context, rather than statistics about it. However, there is still the potential for improved performance through a better model combining existing features, including pattern features.

These experiments have been performed with documents obtained from the retrieval component of the cluster-based strategy. In order to evaluate different answer extraction methods, we assume the answer type is fully known. This way, we decouple question analysis from answer extraction methods. However, both the documents and the keyword extraction and semantic expansion have been performed automatically. In the overall QA process errors propagate from answer type identification down to the answer merging and perfor-

[1] These results are testing only the extraction component, independent from other module. In particular, for the answer extraction-level experiments the answer type is known in advance, hence bypassing question analysis errors. Moreover, because the answer type is known, there is less noise among answer candidates as they are presented to the extractors.

mance decreases. Moreover, more difficult answer types and less dense clusters will not benefit from high quality patterns and will rely on low precisions n-grams.

For the pattern-based extractors, in the context of larger amounts of data and easier to detect answer types, recall is not as important as precision. It is acceptable to miss correct answers in the absence of high precision patterns if sufficient redundancy in relevant documents provides contexts that are more conducive to finding correct answers. A very interesting effect to note is that n-grams that contain question terms but do not contain answers overall perform better than n-grams that must include answers and are focused more in one part of the sentence, but fail to link existing question keywords.

	question level		*strategy level*	
	MRR	*Top5*	*MRR*	*Top5*
keywords	0.431	0.656	0.256	0.461
expanded	0.882	0.914	0.678	0.760

Table 8.8: Comparison between answer extraction using raw question keywords and answer extraction where keywords are semantically and morphologically expanded.

Table 8.8 shows that morphological and semantic expansion of question keywords helps tremendously both in terms of $Top5$ score and MRR score. More patterns are acquired and their precision estimated better after keyword expansion. The expansion of query terms was performed in several stages

1. using a morphological analyzer [84] we obtain multiple forms of the question term. We also use as plural/singular noun forms to expand on original question terms. For proper names, we also expand the term using rules such as: combining the first and last names, using only the last name etc.

2. using the same morphological analyzer, we obtain a simple form of the word (e.g. infinitive in the case of verbs, singular in the case of nouns). Using WordNet [82] we obtain the question term's synonyms. These include different forms for proper names covered by WordNet: e.g. Bob Marley, Bob Nesta Marley, Robert Nesta Marley, Marley.

3. for each of the question term's synonyms, we use morpha and a set of rules for each part of speech to generate multiple forms of the same term. For synonym phrases such as "*pass on*" as the synonym for the word "*die*", we obtain different verb conjugations: e.g. "*passed on*", "*passing on*", "*passes on*"

There are several issues when dealing with this multi-stage expansion of question terms. If a particular proper name is not found in WordNet, it becomes harder to perform morphological and semantic expansion. Furthermore different forms of a question term might be erroneously identified or could be used in different contexts. Moreover, different synonyms (in WordNet synsets) might be more relevant than others given the specific questions. We assign successively lower weights to each expansion of the question term. Moreover, for each synset, we use a decreasing linear function to assign synonym weights. The actual weights and parameters were tuned using a different set of questions, by labeling expanded keywords according to how relevant they were to the original term.

Cluster performance is measured by how well their corresponding strategy can answer new similar questions. As seen in table 8.8, not all strategies are successful. However, since a question belongs to multiple clusters, several strategies are used to find correct answers. If strategy confidence is accurate, answers produced by successful strategies will have a higher score, hence the higher question-level scores.

The **support vector machine-based extractor** is based on correctness classification and can be trained on all features, including proximity and pattern-based features. A major advantage is that it can simultaneously use all proximity functions f_d as well as all pattern features, not only the highest-score patterns. A soft-margin SVM classifier [121, 12, 25] learns by choosing a hyperplane that splits the data points as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples:

$$min||w||^2 + C\sum_{i=1}^{n}\xi_i \quad \text{such that} \quad c_i(\mathbf{w}\cdot\mathbf{x_i} - b) \geq 1 - \xi_i \tag{8.10}$$

In our experiments, we use SVM Light toolkit for classification [59] with a linear kernel.

We maintain a cost-factor of one, such that training errors on positive examples have the same cost as errors on negative examples. Our in-depth experiment dataset contains close to 100 clusters some of which are very broad and cover a large number of question instances. This leads to the training and testing of close to $2,000$ SVM models during the leave-one-out cross validation process for *one* set of parameters and features. Although this is a very large number of SVMs trained, we are taking advantage of every piece of data since other division of training/validation/testing would drastically reduce our modest dataset. The automatically extracted patterns had the most impact on the SVM performance, especially patterns based on question terms, followed by patterns that also included potential answers (based on expected answer type). The next most useful feature class was the lexical items class (actual words). Proximity and sentence statistics were less useful, although helped improve the overall performance.

Integrating the score of cluster-specific classifiers into a global IBQA model is critical in constructing a probabilistic approach. However, depending on the classifier used for different models, the candidate answer correctness scores will be widely different and will not reflect true probabilities. For example in a support vector machines classifier, the goal is to discriminate between two classes +/- and not to assign a true probability that a candidate answer is correct. We transform classifier scores into probabilities by using a classifier re-calibration package [6, 7]. The package uses an asymmetric distribution such as asymmetric Laplace distribution $\Lambda(X|\theta, \beta, \gamma)$ that differentiates between examples that are easy to classify vs. examples that are hard to classify:

$$p(x|\theta, \beta, \gamma) = \begin{cases} \frac{\beta\gamma}{\beta+\gamma} exp[-\beta(\theta - x)] & x \leq \theta \\ & \beta, \gamma > 0 \\ \frac{\beta\gamma}{\beta+\gamma} exp[-\beta(x - \theta)] & x > \theta \end{cases} \tag{8.11}$$

where θ, β, and γ are model parameters: θ is the mode of the distribution, β is the inverse scale of the exponential to the left of the mode, and γ is the inverse scale of the exponential to the right of the mode.

Table 8.9 shows the support vector machine experiment where we combine answer pat-

	MRR	Top5
instance level	0.810	0.824
question level	0.937	0.955

Table 8.9: **Mean Reciprocal Rank (MRR) and Correct in TopK (Top5)**[1] scores for SVM-based extractors using all of the feature classes described above: patterns, proximity, statistics, words. We show MRR/Top5 performance at the instance level and also at the question level, which is more relevant for the overall QA system performance.

terns, question-term patterns, question statistics and proximity features. The classifier takes advantage of the different type of information available in the proximity features to improve on the results of the pattern-based extractor. These results were obtained under ideal conditions where the answer type was already known and the noise was reduced by filtering sentences with a true potential answer (by answer type). In a typical QA system run, recall of potential answer types will not be 100% and more noise will be introduced with potential answers that are not of the correct type. However, to test the performance of the answer extractors it is necessary to create these conditions so that errors from the question analysis and document retrieval do not propagate. These experiments were necessary to evaluate the performance of individual answer extractors under different conditions, independent of the rest of the QA system. This approach is very useful for mult-engine hybrid systems where individual components are brought from different question answering systems and combined into a new QA pipeline. In chapter 10 we integrate the answer extractor and perform experiments with all the components of our instance-based system working together.

In some cases, clusters may have insufficient training data. This situation can occur due to a) the lack of relevant document corresponding to training questions in a cluster, b) a limited number of training questions in a complex cluster or c) the difficult context in which answers occur, leading to poor answer extraction performance. Moreover, the quality of the clustering itself can be low for certain clusters, in which case the corresponding training questions will not be highly similar, and therefore learning cannot occur.

In this case a very simple, high recall and lower precision back-off method is required. When a good extraction SVM model cannot be learned, a proximity-based method can be applied to new questions. Candidate answers are scored according to the number of keywords found in their textual contexts (i.e. surrounding passage) and their proximity to these keywords.

8.2.3 Answer Extraction Scalability under IBQA

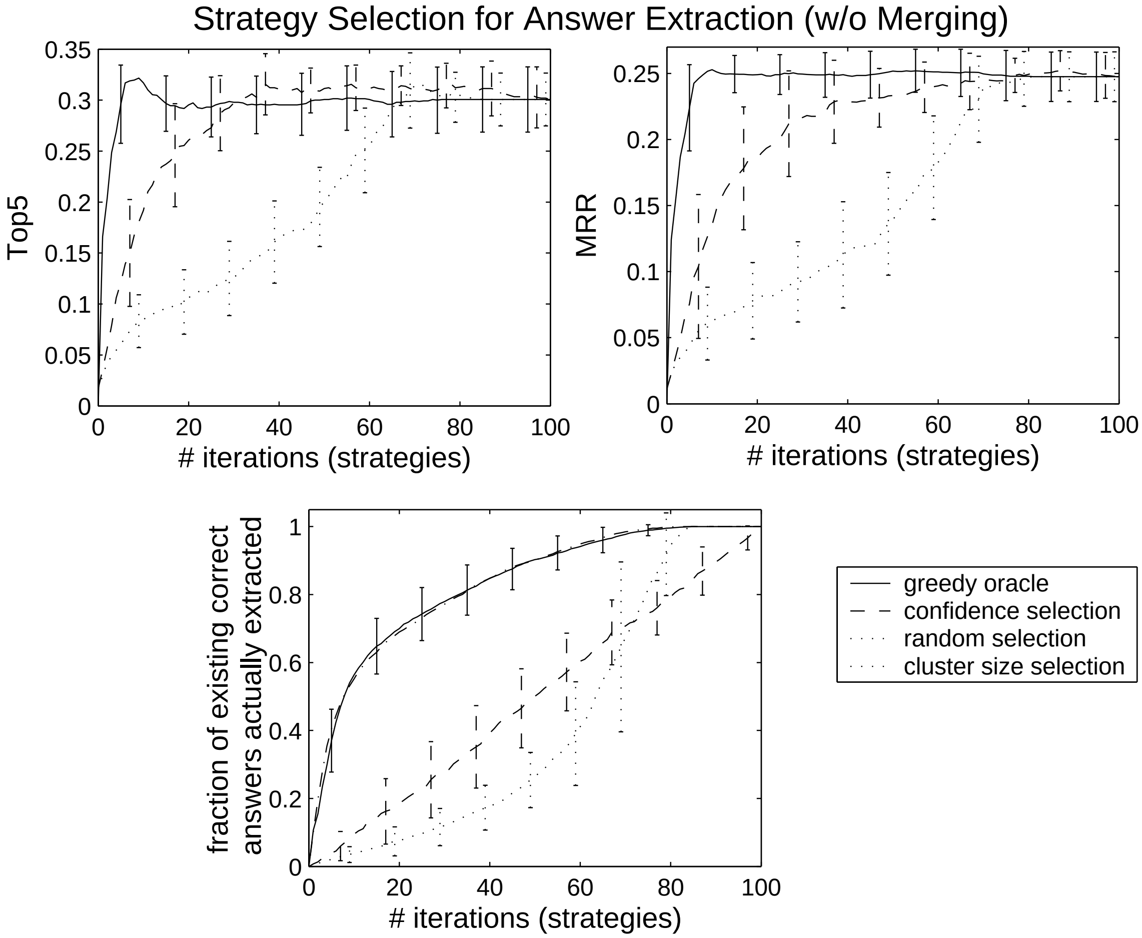

Figure 8.2: Selection based on confidence yields the best performance after carefully selecting a limited number of strategies (cluster, queries, and extractor) to answer a question. However, for redundancy purposes the benefits are more attenuated, and it is better use additional answering strategies if further correct answers are required. No answer merging method has been used here – answers preserve their individual strategy-generated scores.

For a particular cluster, the support vector machine answer extractor is trained on the documents obtained by running the corresponding retrieval component of the answering strategy (i.e. using the learned queries to retrieve documents). The basic features include proximity features, sentence statistics, and patterns (n-grams and paraphrases) that discriminate best between sentences that contain correct answers and sentences that do not. The pattern feature values used in the classifier were not pattern precision or frequency. Instead, they were given by information gain with respect to the positive class (correct answer) and the negative class (incorrect answer) – i.e. the SVM Light classifier uses the feature vectors based on feature selection values, and not frequency or precision.

Under the cluster-based approach, it is not sufficient to train an answer extractor for each answering strategy. These strategies are trained on different number of questions (i.e. cluster size), they are sensitive to the notion of cluster relevance, and they are based on different types of queries and different relevant document distributions. Therefore, the confidence of an extractor has to be taken within the context of its history – i.e. the previous steps in the answering strategy. We measure the answer extraction confidence $conf(A_{AE}(C_j)|q)$ of an answering strategy A derived from cluster C_j given a new test question q:

$$conf(A_{AE}(C_j)|q) = P(a^+|A_{AE}(C_j)) \cdot conf(A_{IR}(C_j)|q) \tag{8.12}$$

where $P(a^+|A_{AE}(C_j)$ is the probability of extracting a correct answer a^+ using the answering strategy $A_{AE}(C_j)$ – more specifically, using the cluster-trained SVM extractor. $P(a^+|A_{AE}(C_j)$ is measured by testing its effectiveness on a held-out set of training questions from the cluster.

In Figure 8.2 we evaluate the effectiveness of our selection method (*confidence selection*) according to MRR, Top5, and the fraction of correct answers extracted out of the total number of correct answers that would be extracted if all strategies were used. The *random selection* consists of randomly sampling from the available strategies and using them to extract more answers. The *cluster size selection* is an intuitive baseline which gives priority to more specific, focused strategies that correspond to clusters with higher similarity to the test question: $P(C_j|q)$. However, it does not perform well, since cluster similarity is a necessary property, but it is not sufficient in the selection process. Finally, the *greedy oracle* optimizes at each iteration the strategy that yields the most additional correct answers. In many cases, our *confidence selection* method performs virtually indistinguishable from the greedy oracle sequence.

While there is a benefit in using this selection method to quickly obtain a larger number of correct answers, if high answer redundancy is required, then further strategies must be used. However, in terms of MRR and Top5 score measures, a very small number of carefully selected strategies can be as effective as utilizing all of the available answering strategies.

A very important observation is that performance does not degrade with subsequent iterations which increase the number of strategies. This can be explained by the fact that the best strategies provide the highest confidence values and corresponding individual answer scores, and unsuccessful strategies do not introduce considerable additional noise.

In these answer extraction experiments for this stage of the instance based question answering, no answer merging method was used. Each instance of an answer was treated separately, with its original confidence score given by a specific strategy. This approach does not provide any boost in confidence if the same answer has been seen more than once. However, it provides a measure of relative answering strategy noise and is informative to the performance of the answer extraction stage in the IBQA system and in general to any question answering system.

8.3 Answer Extraction – Summary

This chapter proposes a cluster-based training of answer extraction models. The answer extractors are trained on abstracted, aggregated cluster-specific data. The training data is based on documents retrieved for multiple similar questions in the same cluster, leading to more focused answer extractor models.

We have shown component level experiments with three different extractors: proximity-based, pattern-based, and SVM-based. The SVM-based extractor performs well consistently, and extractor performance is driven first by patterns extracted from aggregated data and then by discriminative words and proximity features. On temporal data, more than 95% of the questions have at least a correct answer in the top five answers extracted.

Experiments show that careful, but extensive semantic expansion helps the extraction stage in the IBQA system tremendously, almost doubling performance. Moreover, the strategy selection method presented in this chapter obtains very good performance in terms of MRR and Top5 when only 10% of the strategies are selected. However, more strategies help redundancy by being able to extract additional correct answers.

CHAPTER 9

Answer Generation

After question, clustering, document retrieval, and answer extraction, question answering systems are faced with a set of text fragments which are possible correct answers. We shall denote these text fragments *candidate answer components* or *answer candidates* for short. Each of these answer candidates has a history in the pipeline associated with the cluster-strategy selected, the query type used, and the extraction model used. This history corresponds to a set of confidence scores for all of these stages. Often, several distinct answer candidates are identical textual snippets. These candidates usually originate from different passages and are often extracted using different strategies.

Moreover, as illustrated below, these textual fragments may not always constitute full answers[1]. The set of answer candidates obtained through answer extraction could include:

- *complete answer* – text snippet which fully satisfies the information need of the user/analyst,

[1] some complex questions require putting together partial answers from different documents in order to obtain a full correct answer.

unequivocally answering the question:

Question:	*What kind of animal is a corgi?*
Complete Answer:	*dog*

- *overly specific answer* – text snippet with a lower granularity than that of the expected answer, partially satisfying or not satisfying the user's information need:

Question:	*What kind of animal is a corgi?*
Complete Answer:	*canis familiaris*

- *overly generic answer* – text snippet with a higher granularity than that of the expected answer, partially satisfying or not satisfying the user's information need:

Question:	*What kind of animal is a corgi?*
Complete Answer:	*domestic animal*

- *partial answer (factoid)* – text snippet containing part of the information required to answer a question:

Factoid Question:	*When was John Lennon born?*
Partial Answer:	*October*
Partial Answer:	*October 1940*
Partial Answer:	*October 9*

- *partial answer (complex)* – text snippet containing part of the information required to answer a *complex* question (e.g. cause-effect, reason). Partial answers from multiple documents or passages need to be combined (sometimes using inference) in order to generate a complete answer:

Complex Question:	*Why did the dinosaurs die out?*
Partial Answer:	*There is now widespread evidence that a meteorite impact was at least the partial cause for this extinction.*
Partial Answer:	*The two major theories involve (1) gradual climate changes and (2) the collision of an asteroid with the earth.*
Partial Answer:	*Other factors such as extensive release of volcanic gases, climatic cooling [..] may have contributed to this extinction.*

- *partial answer (list)* – one element in a set required by a *list*[2] question:

List Question:	*What countries produce coffee?*
Partial Answer:	*Colombia*
Partial Answer:	*Italy*
Partial Answer:	*Brazil*

In this work we focus factoid questions and where the hard problem is finding a correct granularity for correct answers. Another interesting problem is answer overlap and how to compose a larger correct answer. Dates in particular could benefit from temporal standardization [], but depending on the processing available, merging answers based on n-gram overlap helps consolidate correct answers.

In the example above, some evaluations may deem these candidate answers as correct or incorrect and depending on how answers to questions are assessed. Furthermore, in the TREC evaluation [125], depending on answer specificity, assessors might consider an answer candidate to be exact or inexact – e.g. Paris vs. Paris, Missouri.

Answer generation is the task of taking the output of answer extraction (i.e. candidate answer components) and produce correct and complete answers with corresponding confidence scores. This task involves combining evidence from several answer candidate components in order to generate meaningful correct answers with high confidence scores.

Section 9.1 explores previous work in answer clustering and merging, answer composition and fusion, as well as similar problems encountered in fields other than question answering. Section 9.2 presents an approach to the answer generation problem that we implemented under the instance-based approach.

9.1 Related Work

The MIT system's [47, 60] definitional question answering component integrates results from three sources: database lookup, dictionary lookup, and document lookup. If two an-

[2] considered a special type of complex question

swers share more than sixty percent of their keywords, one of them is randomly discarded. Answers are meta-ordered by expected accuracy of the technique used for extraction. Answers are further ordered according to accuracy of individual extraction patterns (regular expressions) computed on a training set. The answer merging step decides only the number of answers nuggets to provide: n if $n \leq 10$ and $n + \sqrt{n - 10}$ if $n \geq 10$. This strategy is reasonable since the evaluation metric for definitional questions is more biased towards recall than precision. For this type of evaluation, including more candidate answers is better.

In Microsoft's AskMSR system [15, 31, 5], answer composition is performed by extracting unigrams, bigrams, and trigrams from document summaries and sorting them according to frequency of occurrence, weighted by scores associated with query types. The n-grams are filtered using surface-level features and then re-scored according to how well they match an expected answer type. Answer composition is performed by *tiling* n-grams [15] together, assembling longer answers from shorter ones.

Related to answer clustering and merging is *answer fusion* [37], a task that addresses the problem of merging answers or partial answers spread across multiple documents. This notion applies especially to complex types such as: list questions, cause and effect questions, or questions that require inference. These types of questions can be decomposed into simpler questions or extraction tasks. The resulting partial answers can be combined into the overall final answer. Most often, these questions are solved by employing a factoid QA system to answer the simpler questions and applying answer fusion to the resulting answers. LCC [40] shows a practical set steps towards resolving complex questions that include question decomposition and answer fusion. In this case answer fusion is used to identify a single coherent answer from a set of different and potentially contradictory answers.

Another statistical approach to answer merging and selection is incorporated in the CMU JAVELIN question answering system [61], which describes the utility of semantic resources such as WordNet, gazetteers, and the web. The JAVELIN system combines multiple resources for computing the confidence scores assigned to correct answers using weighted linear combinations and logistic regression. The answer pool is populated by three extractors that we developed [93] using different techniques of varying quality that were used to find *location* and *proper-name* answers. The selection method based on logistic regression performed best, improving selection accuracy by 32.35% for location questions and 72% for proper-name questions.

9.2 Answer Generation under IBQA

Under the instance-based approach, we plan to explore several methods of answer and answer confidence merging that take into account estimates of success in terms of question clustering, document retrieval, answer extraction, and answer modeling. The first step is to generate directly comparable scores, regardless of the answering strategy employed, with the goal of generating a unified answer set (list) for each question.

Another issue that is important in answer generation has to do with how similar candidate answers are. After directly comparable scores are generated, candidates in the answer set will exhibit various degrees of overlap and similarity: full overlap (e.g. answer candidate “Mount Everest” occurs four times in the answer set with different confidence score), partial overlap (e.g. “January, 1984”, “January 12”, and “January 12, 1984”), granularity (e.g. “August” and “Summer”), and similarity (e.g. “blood disorder” and “disorder of the blood”).

An existing solution is to examine pairs of candidate answers and assess the degree of *support* that each has for the other and then boost the confidence scores appropriately. Another solution is to cluster the answers and select a cluster representative (answer) and boost its confidence score. For both methods, it is difficult to define a distance function that incorporates similarity, overlap, and granularity. It is also not established how confidence scores are affected when additional similar answers are found.

We explore several methods for merging answers and combining extraction scores. The simplest answer generation approach consists of obtaining the answer extraction scores and considering them comparable across clusters. For each test question, answers originating from different instance of that questions belonging to different clusters are deemed distinct and ordered according to their original extraction scores. More specifically, two answer instances a_i and a_j are treated as distinct answers with possibly different confidence scores even if they consist of identical strings. In reality answer extractors trained on different size clusters and using data from different questions may extract answers whose confidence scores are not normalized or comparable.

An improvement on the previous strategy is to merge identical answers obtained for specific question instances. Since they were obtained using the same answer extractor trained on the same cluster data, the confidence scores are comparable. To compute the confidence $conf_{am}(a_i|C_j, q)$ of an answer a_i to a question q from a cluster C_j, we aggregate the confidence of all answers produced by the cluster-specific extractor to question q:

$$conf_{am}(a_i|C_j, q) = \sum_{a_k \in A(q,C_j)} conf_{ae}(a_k|C_j, q) b(a_i, a_k) \tag{9.1}$$

where $A(q, C_j)$ is the set of answers produced by the answer extractor trained on cluster C_j and tested on question q, and $b(a_i, a_k)$ is a binary function that indicates whether answer instance a_i and answer instance a_k contain the same string.

The previous approach takes advantage of the fact that we can compare confidence scores obtained from the same cluster-specific extractor. However, it does not exploit the fact that the same answer could be obtained via multiple answering strategies. The simplest assumption we can make is that confidence scores are robust across extractors and aggregate over all instances from all clusters. Thus we compute the confidence $conf_{am}(a_i|C_j, q)$ of an answer a_i to a question q as follows:

$$\begin{aligned} conf_{am}(a_i|q) &= \sum_{C_j} conf_{ae}(a_i|C_j, q) \\ &= \sum_{C_j} \sum_{a_k \in A(q,C_j)} conf_{ae}(a_k|C_j, q) b(a_i, a_k) \end{aligned} \tag{9.2}$$

To remove the cross-cluster confidence compatibility assumption, we can modify the previous methods to incorporate a cluster-based weighting scheme where each answer is weighted by the cluster quality and relevance as defined in section 5.5.2. The weighted confidence $wconf_{am}(a_i|C_j, q)$ of an answer a_i to a question q from a cluster C_j is computed as:

$$\begin{aligned} wconf_{am}(a_i|C_j, q) &= \sum_{a_k \in A(q,C_j)} conf_{ae}(a_k|C_j, q) b(a_i, a_k) Q(C_j, q) && (9.3) \\ &= Q(C_j, q) \sum_{a_k \in A(q,C_j)} conf_{ae}(a_k|C_j, q) b(a_i, a_k) && (9.4) \\ &= Q(C_j, q) conf_{am}(a_i|C_j, q) && (9.5) \end{aligned}$$

where we use $Q(C_j, q)$ to estimate the quality of a cluster and its relevance to the test question. We apply the same weights to the question-level merging strategy, where we compute the weighted confidence $wconf_{ae}(a_i|C_j, q)$ of an answer a_i to a question q:

$$wconf_{am}(a_i|q) = \sum_{C_j} \sum_{a_k \in A(q,C_j)} conf_{ae}(a_k|C_j, q) b(a_i, a_k) Q(C_j, q) \qquad (9.6)$$

This approach has the advantage that it merges answers from multiple clusters and combines their confidence scores in a linear combination, using cluster relevance/quality as weights. It has the benefit of compensating for differences in cluster relevance to the test question, but at the same time it might reduce the importance of correct answers extracted from low-quality clusters. We use these merging methods described above and apply them to the IBQA system.

Table 9.1 shows the MRR and TOP5 performance for each of the five methods described above. We use the temporal dataset also employed in previous chapters for component-level evaluation. As inputs to the answer merger component, we use the answers extracted with the proximity extractor on the fully expanded documents obtained during the document retrieval stage. This offers us a good baseline performance, but with the potential for improvement. The *answer extraction* method uses the answer confidences directly from the answer extrac-

	MRR	Top5
answer extraction	0.478	0.641
cluster-level	0.561	0.657
question-level	0.610	0.753
cluster-level, weighted	0.577	0.662
question-level, weighted	0.617	0.758

Table 9.1: **Mean Reciprocal Rank (MRR) and Correct in TopK (Top5)** scores for the answer merging component. We use five merging methods to compute the confidence scores: i) directly using *answer extraction* scores, ii) merging answers obtained with the same strategy, iii) merging answers regardless of the strategy (cluster) used, iv) v) the same as previous methods, but adding cluster-based weights to each answer instance.

tion stage and considers different instances to be independent answers, regardless of whether or not the strings are identical. The *cluster-level* option specifies that we aggregate identical answers for a particular question instance, using only one answering strategy. The *question-level* option specifies that combine answer confidence scores for identical answers regardless of the cluster/strategy used to find them. Finally, the *weighted* methods use cluster-specific weights in a linear combination to combine answer confidences.

We observe that question-level methods are more effective than cluster-based methods which is encouraging. The noise originating from confidence score incompatibility for different strategies has a lower impact compared to the benefits of having multiple strategies finding the same answers. Also, by using a linear combination of answer instances, with non-uniform weights, we further obtain a small performance improvement. Compared to extraction confidence scores, using an answer merging component has a significant benefit in our instance-based QA system, obtaining an performance improvement of 29%.

9.2.1 Strategy Selection for Answer Merging

Experiments with answer merging under the instance-based approach include a closer look at strategy selection optimized specifically for this particular stage in the QA process. More specifically, cluster-based strategy selection involves combining known confidence information from cluster quality, document retrieval and answer extraction stages to optimize for overall question answering system performance using an existing answer merging method.

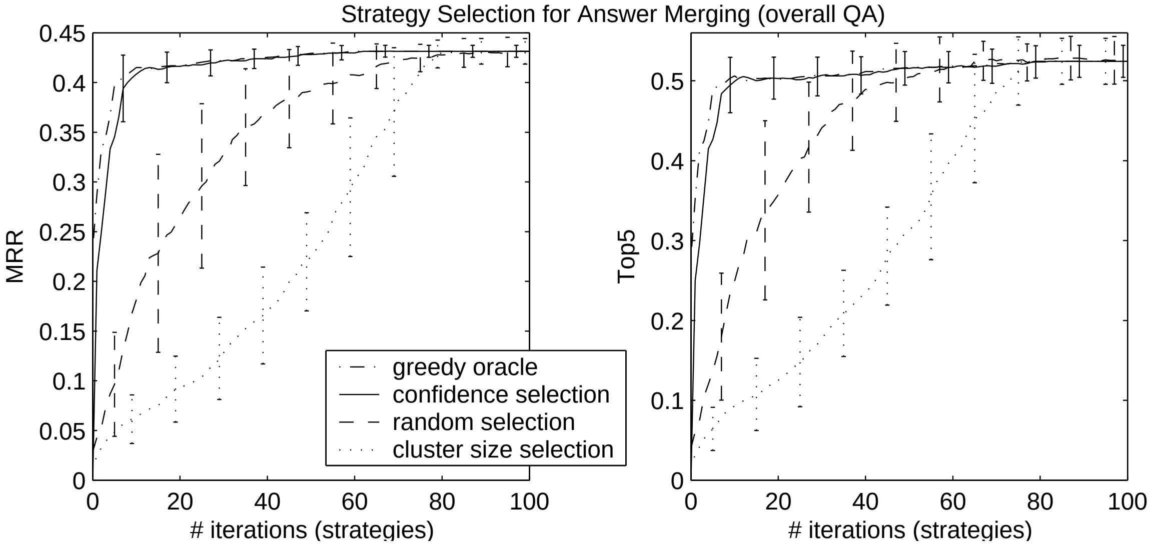

Figure 9.1: Final QA performance – for answer merging, confidence based selection performance allows the QA system to select less than 10% of the strategies with nearly maximum performance. The trade-off is marginally better for the MRR metric, since it requires better relative scoring from answer merging.

At the answer merging stage in the question answering pipeline, multiple answering strategies were *activated* by selecting clusters of similar training questions and applying their chain models to a new question. Each of these clusters leads to a cluster-specific answer set: the cluster-based strategy goes through document retrieval and answer extraction, generating a set of answers with confidence scores. The task of the answer merging component is to make use of redundancy and re-score the answers such that the correct answers are ranked higher than incorrect answers. The answering merging method we implemented consists of a weighted sum of the individual answer confidences for all answer instances with the same surface form. The answer confidence $conf(a_k|q)$ of an answer a_k at the end of the question answering process is aggregated across all clusters C_j and is given by:

$$conf(a_k|q) = \sum_{a_k} \sum_{C_j} P(a_k|A_{AE}(C_j)) \cdot conf(A_{AE}(C_j)|q) \tag{9.7}$$

where $P(a_k|A_{AE}(C_j)$ is the probability of extracting a correct answer a_k using the answering strategy $A_{AE}(C_j)$.

Both in terms of MRR and Top5 scores, as seen from Figure 8.2 and Figure 9.1, the weighted answer merging method gains approximately 0.15 MRR points (60%) and also 0.15 Top5 points (43%) in performance. The gap trade-off between using the *confidence selection* scores and using all strategies also improved. As in the case of answer extraction, it is encouraging that the *confidence selection* approach closely follows the *greedy optimal* selection. It is important to note that greedy optimal selection is based on selecting the next strategy that is known to be the best. However, an overall optimal strategy selection would explore all combinations of strategy selection sequences.

CHAPTER 10

End-to-End IBQA Experiments

10.1 Experimental Setup

One of the critical issues in training a statistical system is data availability. For the instance-based question answering approach in particular, we rely on question-answer pairs as the raw data from which we derive answering strategies. The question collections we are using in our experiments are question datasets used for the past TREC evaluation (TREC 8-13). These collections cover mostly open-domain factoid, definitional and list questions. In this work we focus on questions whose answers are factoidal in nature – the answer is often expressed as a very short noun phrase, sometimes even one-word long. We treat definitional questions as a different class of questions and we show the instance-based approach applicability to a set of definitional questions.

Since the task we are addressing is open-domain question answering, most of the questions have corresponding correct answers on the web. Moreover, in most cases, the web has a high data redundancy and more specifically correct answer redundancy, that the instance-based QA approach can take advantage of. We use the web as the main data source for relevant documents, but also for large amounts of training data that have the potential to lead to high precision extraction models. Many systems participating at trec find answers using the web and then project these answers onto a local corpus that is provided to all TREC participants: the TREC and AQUAINT corpora. We also use these local corpora for extracting question-answer pairs and show overall QA performance improvements.

The TREC and AQUAINT corpora consist of news stories from various sources. The TREC corpus contains one million news documents (approximately three gigabytes) from Associated Press (AP) newswire, Wall Street Journal, San Jose Mercury News, Financial Times, Los Angeles Times, and the Foreign Broadcast Information Service. This corpus was used for the first years of question answering evaluation performed by NIST.

The AQUAINT Corpus of English News Text (LDC catalog number LDC2002T31) consists of AP newswire 1998-2000, the New York Times newswire (1998-2000), and English documents from Xinhua News Agency (1996-2000). It contains three gigabytes of text representing more than a million documents. This corpus has replaced in recent years the TREC corpus as the target data source in NIST's question answering annual evaluation.

The Web is very often used as a corpus in question answering research. Some QA systems have been developed specifically for the Web[1]. They differ from systems focusing on local corpora since they use dedicated search engines that very often find relevant documents. In addition, due to the redundancy of information, many more documents are likely to contain the desired information. Moreover, answers on the Web are likely to be found in very simple contexts and be much more easily extracted than answers in a local corpus. Therefore, the retrieval and extraction steps in the QA process have the potential to perform much better.

Dedicated structured sources such as encyclopedias (e.g. Wikipedia, Encyclopedia Britannica), dictionaries (e.g. Webster, The Free Dictionary), specialized websites (e.g. biog-

[1] many systems obtain answers from sources other than local corpora and then project them and find similar answers into the local corpus.

raphy.com, 50states.com) and gazetteers (e.g. CIA World Factbook), as well as resources such as WordNet are also used in question answering. They constitute additional structured sources for which specialized queries and extraction are performed in order to extract focused answers for particular types of questions. While very precise, these sources create a strong dependency of QA systems to question types, resources and query expansion and answer extraction expert rules. Although we will not focus on these resources as a main source of answers, we will explore the usefulness of incorporating such sources into the IBQA system.

The TREC question collections contain factoid, definitional, and list questions from AOL and MSN Search logs. Some questions have no known correct answers in the TREC and AQUAINT corpora. Furthermore, for a text segment to constitute a correct answer (i.e. 1984) to a question (*What is the title of George Orwell's best novel?*), the context has to actually answer the question (*his novel "1984"*), rather than mentioning it in another context (*1984 was a year of great unrest*).

In chapter 11 we show how additional QA data can be acquired through semi-supervised methods. The acquisition of high quality question-answer data is still a subject of current research. However, we show that using semi-supervised techniques, we can target the acquisition for a subset of the instance-based QA clusters that contain highly focused similar questions. We show that performance increases with more data and we directions for future research that can benefit the IBQA approach, as well as other statistical QA methods.

Data sparsity is certain to be a very important problem in training statistical question answering systems. Under the instance-based question answering approach, some clusters have sufficient data for deriving answering strategies, while other clusters either do not have sufficient data, or are not be cohesive enough to produce good models. When merging and scoring answer candidates, answers generating from strategies based on sparse clusters or low confidence models do not obtain a high score. We show that if sparse clusters of highly similar questions do not perform well, acquiring question-answer pairs of similar questions results in better models, and better cluster-specific models. It is not within the scope of this thesis to acquire sufficient amount of data to outperform state-of-the-art QA systems that benefit from years of system engineering, deep question and document processing, and high quality knowledge resources. We show that a core resource non-intensive IBQA approach with limited available data constitutes a very high domain and language independent baseline. Moreover, we show that additional training data closes the gap between such systems and statistical approaches.

As mentioned in section 4.2.5, the evaluation is performed using MRR and Top5 scores based on automatic correctness evaluation. Answer correctness is done using answer patterns (regular expressions) that have been built based on system outputs and local corpus search. Due to the limited domain of these candidate answer sources, the answer patterns used in the automatic QA evaluation have small coverage, not accounting for:

i) the existence of additional different correct answer – e.g. if “1984” is a correct answer to the question *What is the title of George Orwell’s best novel?*, another correct answer could also be “Animal Farm”.

ii) surface form variability based on morphology and syntax – this problem accounts for a large number of mismatches between TREC-based answer keys and web-based answers (e.g. American vs. America)

iii) semantically equivalent answers with minor or no surface-form overlap – e.g. “1984” vs. “nineteen eighty-four”. These answers are often based on paraphrasing (e.g. “Nobel laureate” vs. “Nobel prize winner”).

For these reasons, when the retrieval component of a QA system is web-based, the relevant documents obtained may offer a greater variety of correct answers that may not be covered by existing regular expressions. At the same time, web documents are typically noisier and may disrupt the extraction model being learned. However, compensating for the noise, the web tends to offer a greater redundancy of correct answers.

10.2 Factoid Questions

Questions whose answer are simple, factual answers, that can typically be succinctly answered using a small (noun) phrase are called *factoid questions*. This class includes questions whose answers are locations: dates, people's names, objects, numeric expressions with and without units, etc:

Sample Factoid Questions	**Correct Answers**
Where is London?	Europe; England
What city was the host of the 2005 Olympics?	Turin; Torino; Italy
Who was Clinton's vice president?	Gore; Al Gore
Who was the first person to step on the moon?	Armstrong
What kind of animal is a corgi?	dog; canine
What is the most popular product of Belgium?	chocolate
How tall is Mt. Everest?	8,850 M (29,035 feet); 30,000 ft
How big is Australia?	7,682,300 SQ KM; Pop: 20,090,437
...	

Document density is highly dependent not only on the retrieval engine used but more importantly on the actual corpus form which the documents are obtained. Figure 10.1a) shows that for web documents, we obtain a sustained minimum relevant document density of above 0.2 for each rank. This means that more than 20% of the questions have a corresponding relevant document for each rank $1 - 100$. This finding is encouraging since it means that retrieving more documents from the corpus is beneficial – i.e. we can obtain more relevant documents. For each corpus used, the relevant document density will be different and training a specific retrieval limit should be performed for each. Moreover, a planner for question answering could take advantage of this distribution and dynamically adjust the number of documents retrieved.

A useful quality we look for in a retrieval QA component is for documents ranked higher to be more likely to be relevant. Towards this end, we measure the mean average precision

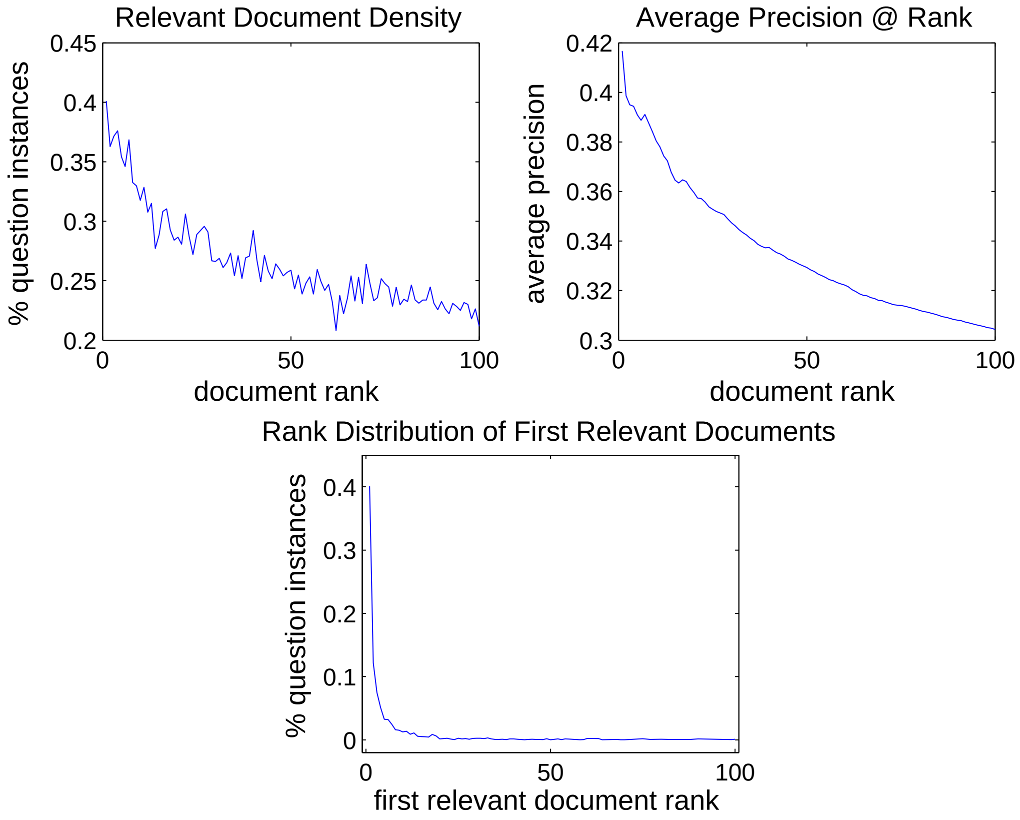

Figure 10.1: Retrieval characteristics under the instance-based approach: a) the distribution of relevant document density over ranks, b) average precision at number of documents retrieved, and c) the rank distribution of first relevant document.

over the set of questions where the average precision A_p is the average of the precision P after each document retrieved d_r:

$$A_p = \frac{\sum_{r=1}^{n} P(r) \cdot B(r)}{\sum_{r=1}^{n} B(r)} \tag{10.1}$$

where $B(r)$ is a binary function which is 1 if the document is relevant and 0 if it is not, and n is the number of documents. Average precision puts a greater emphasis on ranking relevant documents higher. We measure the mean average precision of our factoid question retrieval component 10.1b). Results show that average precision does indeed follow the desired principle: higher ranked documents are more indeed relevant than lower ranked documents.

Another performance metric for question answering retrieval components measures how many non-relevant documents one needs to retrieve before obtaining a relevant document. This metric is not a good measure of relevant document density or retrieval precision. In the context of question answering however, the rank of the first relevant document retrieved (figure 10.1c) can be used to learn the minimum number of documents that must be retrieved to obtain at least a relevant document. This is especially useful in an interactive environment where user utility needs to be optimized for. This metric tests the combination between retrieval engine performance and easily accessible relevant data in the corpus.

	MRR	Top5	Top1
instance level (precision)	0.435	0.511	0.356
question level (precision)	0.5	0.584	0.434
question level (overall)	0.432	0.496	0.375

Table 10.1: Mean Reciprocal Rank (MRR) and Correct in TopK (Top5) scores for IBQA using all of the features above. We show MRR/Top5 performance at the instance level and also at the question level, which is more relevant for the overall QA system performance. In these experiments we assumed perfect answer type classification. We show the overall QA performance (on all test questions) as well as the precision, which is the performance on questions the system attempted to answer.

Table 10.1 shows the MRR and Top5 scores for answer extraction for the overall QA system, trained on all types of questions available from official evaluations TREC8-12. The small difference in MRR and Top5 scores indicates that for most questions, the first correct answer corresponds to a high rank. More specifically, our IBQA system obtained the Top1 score of 0.384 at the question level, which corresponds to considering only one answer for every question (i.e. TREC score). Furthermore the score difference between instance and question level indicates that more than one cluster typically produces correct answers. However, there are cases when answering strategies based on certain cluster do not generate correct answers – hence the difference in performance between instance level and question level.

In an instance-based question answering, the precision is measured by computing a metric on the questions for which there is sufficient training data. More specifically, certain questions are not included in any clusters of similar training questions and therefore the system cannot apply any answering strategy to try to answer the questions. The overall performance is computed on all test questions regardless of whether the QA system has any strategies that can handle them or not. The overall performance is usually what QA systems report on for official evaluation data such as TREC and CLEF. However, there is a benefit in computing the precision scores since this gives us more insight into the problems of the system and into what type of additional training data is needed to improve the QA system further. Our IBQA system implementation obtains a high precision both in terms of MRR and TOP K correct metrics, suggesting that there is a subset of questions in the TREC question set that that are outliers – do not have other sufficiently similar questions in the same question set.

The instance-based question answering approach greatly benefits from large amounts of data as well as from redundancy in the document collection. In our implementation, we used the Web as our corpus and thus improved our chances of finding relevant documents. It is also more likely to find a higher number of correct answers in extraction-conducive contexts in web documents compared to a limited local corpus. However, the models learned form online data could also be applied to local corpora, thus benefiting from high redundancy in the training data. A negative impact of using web documents occurs due to the transient nature of certain correct answers to questions that depend on how current relevant documents are. Furthermore, the wide variety of answer forms on the web is not captured by the answer keys available for TREC questions.

Table 10.2 shows the results of the top 10 systems participation at TREC evaluations in each particular year, as well as the performance of the instance-based question answering system both in terms of MRR and Accuracy – i.e. average accuracy of the top1 answer for every question. Since the QA field evolved over the years, there are many reasons why the evaluation results from year to year are not fully comparable: different evaluation metrics, different definition of an answer, question-interdependence, the presence of within-question and cross-question references, as well as different human assessor guidelines.

For the first two TREC question answering evaluations, participating systems were allowed to use answer snippets of either 50 or 250 bytes (two different tracks). The metric of

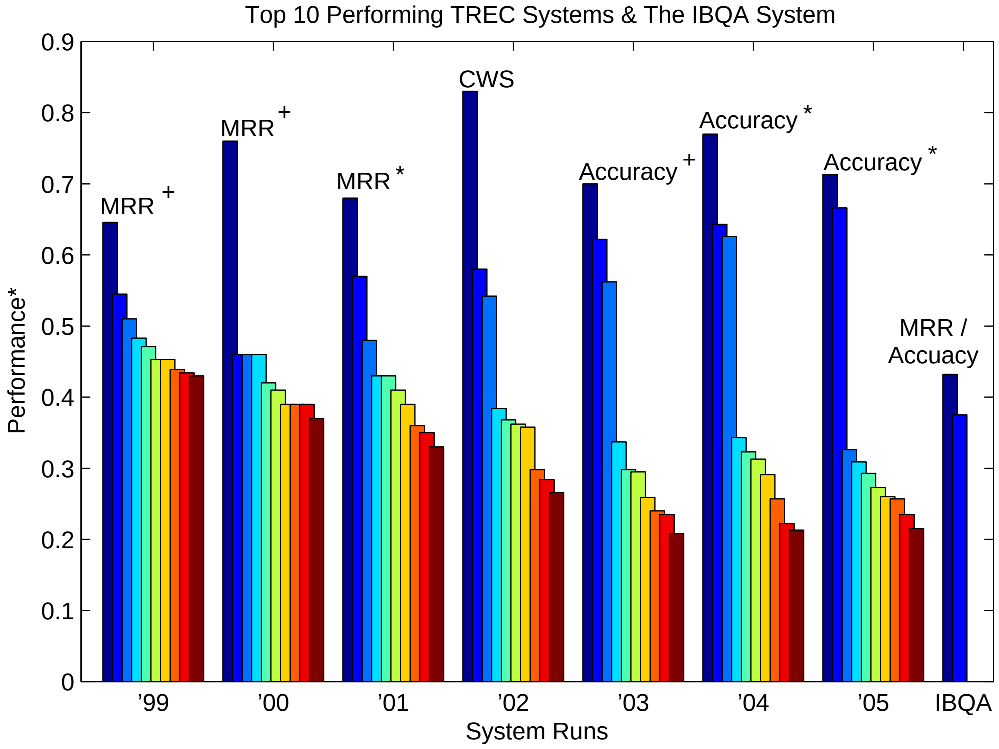

Figure 10.2: Top 10 performing systems participating in the official TREC evaluation different years and the IBQA system performance. The first two TREC competitions were evaluated using MRR on answer segments of 250 bytes. TREC 2001 was evaluated using MRR on answer segments of 50 bytes. TREC 2002 was evaluated using CWS on answer exact answers. TREC 2003 was evaluated using simple accuracy (top1/percent correct). TREC 2004 and 2005 were evaluated using accuracy (top1/percent correct) on inter-dependent questions. For IBQA we present the results using two performance measures: the MRR on exact answers and the accuracy (top1/percent correct).

choice was mean reciprocal rank (MRR) over the whole set of questions. Observing that the 250 byte track yielded much better overall results due to the wide coverage of the answer snippet, in the subsequent year 2001 the focus was only the 50 byte track. In 2002 a new measure confidence-weighted score (CWS) was used in the TREC question answering evaluation, which focused on the top 1 exact (as opposed to snippet) answer. CWS scores take into account the individual answer confidences, re-ranking the questions accordingly.

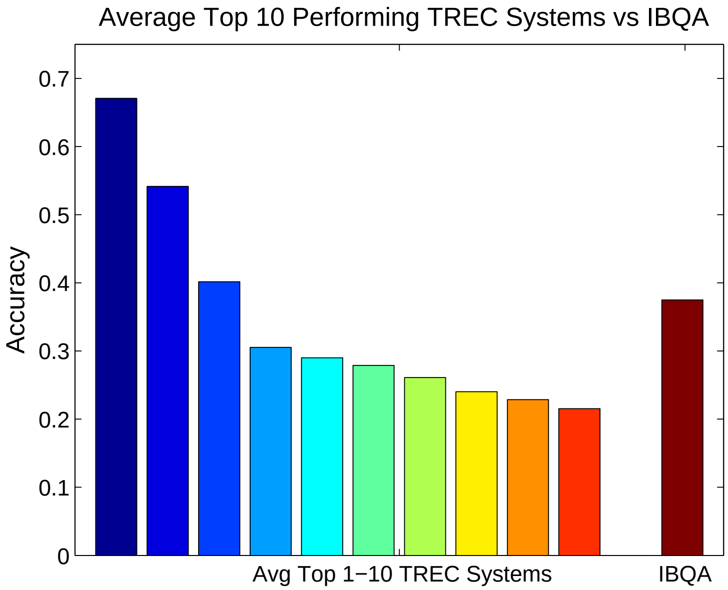

Figure 10.3: Average of the top 10 performing systems participating in the official TREC evaluation and the IBQA system performance. We overestimate the results of systems at the first three TREC competitions by using MRR for Accuracy. Also, for TREC 2002, we obtained the Accuracy scores rather than the CWS scores from the official evaluation.

Finally, simple accuracy for exact answers was used in the TREC 2004 and 2005 evaluations, also considering only the top1 answer returned by each question answering system. This metric rewards systems that answer questions correctly using only the top answer, but penalizes systems that perform well consistently (e.g. most correct answers are in the top five answers, but not the first). Moreover, starting with 2003, the questions used in the TREC evaluation were inter-dependent, using a shared target event or entity and accepting within-question and cross-question references (nominal, named, and pronominal): e.g. i) Target: *Nirvana* ii) Question 1: "*Who is their lead vocal?*" iii) Question 2: "*When did he die?*".

Since it is data driven, we maximized in the IBQA system the use of all the questions by performing leave-one-out experiments – i.e. train on $n - 1$ of the available questions and test on the remaining one. The IBQA system achieved good performance on TREC questions both in terms of MRR (0.432) and in terms of Accuracy (0.375). These results are very encouraging, especially since the IBQA system is not resource-intensive, does not rely on manual rule engineering or other human expert-heavy components, and can be implemented and maintained with little human involvement. On the other hand, most question answering systems are developed and improved over much longer periods through the effort and expertise of multiple people. Most systems are also resource intensive and the interaction between the main components and the resources is hard-coded and not standardized.

We observe (Table 10.2) that our system performance is a little below the average of the third system at TREC and well above the performance of other systems. The average TREC scores are slight overestimates since we used the MRR scores as replacements for Accuarcy. However, with the IBQA system the answers were not projected back into the documents in local TREC and AQUAINT corpora. To make the comparison as fair as possible, we used the 50 byte window answer definition from TREC-8 and TREC-9 rather than a 250 byte window.

Ideally, during official evaluations systems could be measured with and without question inter-dependence that imposes a very restricting reference resolution burden on QA systems. Currently, only a handful of systems achieve good performance on the task of answering single questions, while the results are not easily replicable due to the complex and tailored integration of various methods and resources. To avoid additional effort on the part of assessors, two different tracks with and without question inter-dependence could use the same questions, permitting question answering systems to attempt to solve first the problem of answering single questions correctly, using methods whose results are **replicable**. The IBQA approach offers a robust question answering platform that is more amenable to experiment replicability, by virtue of being data-driven as well as much easier to develop and train.

10.3 Definitional Questions

During recent years, evaluation forums such as TREC [126] and CLEF [76] have stimulated a tremendous growth of the question answering (QA) field. Successful complex architectures [42] incorporate elements such as statistical components [69, 56], knowledge resources, answer verification, planning, and theorem proving. The main thrust in these evaluation forums has been solving *factoid questions*, questions that accept factual answers (e.g. "*In what year was the first AAAI conference held?*", "*Who was the AAAI chairperson in 1999?*"). We have used factoid questions in most of the experiments performed with the instance-based QA approach. These questions require concise answers representing simple *factoids*: e.g. person names, dates, objects etc.

Another class of questions being explored in the question answering community consists of definitional questions. Definitional questions seek to define entities such as objects, "*What is ouzo?*", concepts "*What is artificial intelligence?*", and people "*Who is Turing?*". Answers to definitional questions are usually longer and more complex. For each entity there can be multiple definitions addressing different aspects of that entity. Most of the definitions considered in official evaluations are also factual in nature and are meant to satisfy the user's factual information needs. QA systems that can successfully answer definitional questions [133, 47, 11, 102] use both structured resources (e.g. WordNet, Wikipedia, Merriam-Webster) and unstructured data (e.g. local corpora, the web) for fact extraction.

10.3.1 Related Work

The TREC 2003-2005 evaluations have been re-structured such that they included factoid questions grouped around a set of target entities. For example, for the target entity "*Franz Kafka*", associated questions included: "*Where was he born?*", "*When was he born?*", "*What books did he author?*" etc. This current TREC evaluation format aims to explore individual factual aspects of a particular entity, similar to answering a definitional question, but using more focus questions. A related and popular information extraction task similar to the definitional QA task is the construction of entity profiles [63] and entity modeling [**?**]. Properties and relations such as (age, affiliation, position, modifiers, descriptors) involving target entities are extracted from raw text in order to build an entity profile.

Due to the formulation of existing QA tasks, definitional question answering systems strive to satisfy the need for factual information. In the process of answering definitional questions, such systems filter out non-factual information, as well as marginally factual information that does not fit into a predefined view of what a definition should be. However, it is often the case that entities (e.g. people and objects) exhibit properties that are hard to capture by standard factual methods. Moreover, there are qualitative attributes and specific factual information often associated with entities that are not captured by existing QA systems. These qualitative elements tend to complement factual data and satisfy a different kind of information need associated with definition questions. In [68], we have have taken a closer look at qualitative aspects of definitional questions that go further than incipient opinion-based question answering work [114] done in conjunction with question answering. Although not the focus of this work, an instance-based system can be adapted to finding qualitative answers to definitional questions. In the context of IBQA, the document sources can be specified to include more qualitative sources: e.g. newsgroups, discussion forums, and blogs. Furthermore the answer set obtained from such sources can be improved by filtering out factual answers obtained from structured sources.

TREC 2003 F$\beta = 5$	0.555 (BBN)	0.473	0.461	0.442	0.338	0.318
TREC 2004 F$\beta = 3$	0.460 (NUS)	0.404	0.376	0.321	0.307	0.285

Table 10.2: Top performing question answering systems in the 2003 and 2004 TREC definitional ('other' category) evaluations. The scores are not directly comparable since the two evaluations had different question distributions and they were computed using different values for the F-measure β parameter.

Several systems extract answers to definitional questions from structured sources. BBN's approach to definitional question answering [133] is based on extracting linguistic features from raw text and then ranking them against a question profile developed using web resources. The features used in this work include appositives, copulas, structured patterns, relations, and propositions. The factual profile was generated from WordNet glossaries (*www.cogsci.princeton.edu/ wn/*), the Merriam-Webster dictionary (*www.m-w.com*), the Columbia Encyclopedia (www.bartleby.com), Wikipedia (*www.wikipedia.com*), and biographies from (*www.s9.com*). TREC 2003 experiments (Table 10.2) produced a 0.555 F-measure$_{\beta=5}$ score and showed that using the Rouge metric [65] to score the answers obtained by BBN correlate well with subjective evaluation results. At the TREC 2003 evaluation, the National University of Singapore system obtained the best performance on definitional questions (i.e.

the *other* category) with an F-measure$_{\beta=3}$ score of 0.460. The system implemented a bigram *soft* pattern model [27] to identify good definition sentence candidates.

A multi-strategy approach to definitional questions was MIT's system [47] which employed a database of surface patterns constructed offline and a web-based dictionary. This simple approach shows good performance in the TREC 2003 evaluation with an F-measure of 0.3 (where $\beta = 5$). An important contribution of this work is the fact that it includes component performance analysis and error analysis with respect to the TREC scoring metric.

Most of the current work on definitional questions has been open-domain question answering – meaning that most of the questions asked in official evaluations have been based on generic web logs and have answers in news stories. Many QA systems have been tailored to these datasets and have implemented interfaces with specific resources such as *biography.com* or Merriam-Webster. This customization makes it harder to port them to new domains. Recent work has been done in evaluating question answering components in the medical domain, in particular, search engine evaluation for definitional questions posed by physicians [135]. More effort is required to successfully apply existing techniques for answering open-domain definitional questions to particular domains.

10.3.2 Experiments

Definitional questions are more difficult to evaluate than factoid questions in that definitional answers are more complex and can be expressed in a very large number of different forms. In particular, since answers are longer (e.g. longer phrases or sentences) paraphrasing, the use of different surface forms, syntax and semantics play a much more prominent role than in answers to factoid questions. Moreover, as shown before, answers to definitional questions can focus on different facets of the concept, or entity they are describing, adding to definitional question complexity – e.g. all of the following definitional aspects can be simultaneously true: Nobel Peace Laureate, archbishop, South African, master of theology, professor , author and husband.

For the definitional question experiments we clustered both according to answer type as well as to surface form. The features include part-of-speech tags, unlimited size n-grams (extraction of which is feasible due to the limited number of words in a question) and paraphrases, as well as casing information (e.g. lowercase vs. uppercase). Similar to many systems participating in the TREC evaluation we retrieved documents from the web using the google api (*www.google.com/api*).

Dataset	*MRR*	*Top5*
Object definitions	0.419	0.582
Person profiles	0.589	0.646
All definitional questions	0.457	0.596

Table 10.3: MRR for object definitions (i.e. *what*) and person biography (i.e. *who*) questions. For this experiment, an answer is considered correct if it contains at least one vital answer nugget. The goal is to understand how often answers presented to the user actually do contain at least a minimum of correct and pertinent information.

Nugget-based automatic measures for definitional QA performance attempt to take into account answer utility from the perspective of human users by favoring recall versus precision (e.g. in TREC evaluations). However, for human users recall alone is a very good measure of performance when answers are very short. This means that it is acceptable for precision to be lower if the answer is short enough – e.g. a seven-word answer that contains a two-word vital nugget. In Table 10.3 we show the performance of our system in terms of MRR and the Top5 scores computed based on recall. These metrics first identify the presence of correct vital nuggets in answers and then compute the MRR and Top5 metrics regardless of whether additional content appears in the answer candidates. The answer candidates were limited to ten words each to decrease the importance of precision.

Object definitions and person profiles are two of the most common categories researchers report on. Since there are research efforts focused on person profile extraction outside of question answering, this class of definitional questions can draw on specific extraction-based methods such as IBM's *QA by dossier with constraints* method [100]. Table 10.3 shows the considerable performance difference between object definitions and person profiles. One explanation is that very often, in news stories, people are referred in the <*function*><*name*> form (e.g. "*archbishop Tutu*"), where <*function*> is either a profession, or a title considered vital in the description of that person. Common entity, object,and concept definition nuggets appear less often in such contexts and more often in more complex syntactic and semantic structures. Another reason for this discrepancy is that object definitions are a catch-all category and is less well defined compared to person profiles. In table 10.3 we also show the overall MRR and TREC performance (micro-average) on all definitional questions.

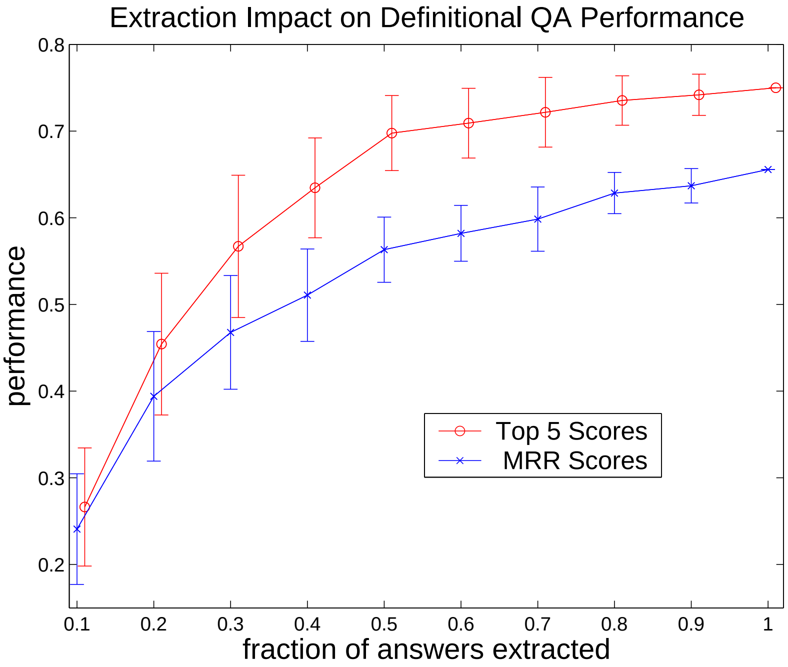

Figure 10.4: Definitional question performance as a function of the fraction of extracted answer. We vary the answer extraction stage output by randomly filtering out sentences from the relevant document pool.

An important consideration in our instance-based framework is how performance varies with training data size. In particular, we are interested in the cases where retrieval and extraction under-perform or there is not a sufficient amount of data to train good models. Figure 10.4 shows recall-based MRR and Top5 performance as we vary the number of relevant sentences that are retrieved, and therefore the number of correct answers that are extracted. We randomly remove sentence/answers from the available data and observe performance differences. As expected, with more relevant sentences and therefore correct answers, the definitional question answering system performs better. However, it is interesting to notice that with half of the data available, MRR and Top5 performance is 80% of the maximum system performance.

F-measure	*Character-level*			*Word-level*		
Dataset	*F1*	*F3*	*F5*	*F1*	*F3*	*F5*
Object definitions	0.275	0.298	0.303	0.291	0.304	0.306
Person profiles	0.376	0.416	0.425	0.386	0.420	0.427
All definitional questions	0.301	0.329	0.335	0.316	0.335	0.338

Table 10.4: Character-level and Word-level F-measure performance for definitional questions. We compute the TREC-based scores for object definitions and personal profiles/biographies using different precision-recall balance parameters that have been previously used in official evaluations.

We compute the nugget F-measure scores for all definitional questions together (equation 4.6), as well as for the two categories previously considered. We consider nugget recall R_{def} to be the ratio of the matched nuggets to the total number of vital nuggets and nugget precision P_{def} to be based the ratio between answer length $|a_i|$ and answer allowance length L_a. Answer and answer allowance length can be computed either at the character level (e.g. number of characters in a string) or at the word level (e.g. number of words/tokens in a string). We observe that for this dataset both length measures maintain the same relative object-person-all definitional question performance.

We are also interested in observing the differences in F-measure performance when we vary the β parameter. In our experiments we used the values of β that have been used in past TREC evaluations. Since experiments are evaluated automatically, we rely on the assumption that these definitions of recall and precision approximate true nugget precision and recall. These results demonstrate that the instance-based approach can be applied to factoid questions as well as to definitional questions. Due to the differences in train/test datasets and the lack of question-level system output for TREC participating systems in the definitional track, the results are not fully compatible to the official full definitional TREC evaluation. However, in our experiments we used definitional question datasets from official TREC evaluations for training and testing, official answer patterns and nuggets, as well as the F-measure-based metric developed by NIST.

Qualitative Exploration

Similar to experiments using factoid questions, we were surprised to observe the automatic discovery of useful and strong data sources for specific question types/clusters during the retrieval stage. For example websites that focused on person biographies and profiles generated generalizable query content to be applied to new questions. Content specific to *biography.com* and *en.wikipedia.org* was incorporated into queries. The online version of wordnet was a good source for answers of object definition (i.e. what) questions.

Question: Who is Anubis?
Vital answer nugget: Egyptian god?

Rank	Strict Judgment	Answer Fragment
A_1	×	*the Patron god of embalmers*
A_2	×	*a god of the dead who is shown as a jackal*
A_3	×	*a man with a head of jackal*
A_4	×	*god of the dead and mummification*
A_5	×	*lord of the Necropolis*
A_6	×	*god of embalming*
A_7	×	*protector of the deceased*
A_8	×	*guardian of the cemetery*
A_9	$\surd$	*jackal-headed Egyptian god of tombs*
A_{10}	×	*conducted dead to judgment*

Table 10.5: Example of answer coverage for definitional questions (actual IBQA system output). Automatic methods may not be able to allow many of these answers to contribute to the correct answer set.

Among the limitations of automatic evaluation of QA system output and in particular with evaluation of definitional questions are reduced coverage of correct answers as well as user bias – i.e. different users might find different definitional answers nuggets to be relevant and/or vital. Consider the question in table 10.5 with its corresponding extracted candidate answers. Even though each of these answers describe at least a definitional aspect of Anubis, only A_9 is considered correct by the automatic evaluation. Some of the answers include *occupational* descriptions (e.g. "patron of embalmers", "protector of the deceased","guardian of the cemetery"), others the core essence (vital nugget) of Anubis (e.g. "god", "lord"), and yet others a more description-oriented aspect of his profile (e.g. "jackal-headed", "shown as a jackal”).

Under a strict answer nugget matching, only “Egyptian god” consists of a vital answer, while an answer simply covering the word “god” is inexact. Moreover, different users might agree on different aspects to be included in the definitional profiles depending whether they are Egyptologists, anthropologists, or middle school report writers.

Beyond clear definitional aspects, non-factual answers such as “Anubis was a perfectionist” or “Anubis is a cold individual” pose a problem to question answering systems. Sentimental analysis work and opinion classification for question answering is starting to make use of such statements. Furthermore, secondary answers that are indeed factual in nature may derail a question answering system by providing sufficiently redundant and strong support – e.g. it What is platinum?: platinum as a *metal*, but also as a *type of visa card*.

CHAPTER 11

Question Answering Data Acquisition

Contributions: *The instance-based question answering approach as well as other statistical components in QA rely on the growing amount of data available for training. Towards automatic methods of data acquisition, this chapter introduces a new, semi-supervised method for acquiring QA data for questions based on semantic relations.*

Data-driven approaches in question answering are increasingly common. In recent years, question answering components in that incorporate learning methods have started to be more frequent [19, 106, 32]. Although the field is still dominated by knowledge-intensive approaches, components such as question classification, answer extraction, and answer verification are beginning to be addressed through statistical methods. Moreover, our recent research [69] shows that it is possible to successfully learn answering strategies directly from question-answer pairs through an instance based approach.

Question analysis and classification components using statistical learning, often require that each category (or type) in an ontology cover a minimum number of question instances, in order to derive strong models. Larger datasets are required for data-driven systems to be able to accurately make use of these ontologies. Since availability of training data for such approaches is very limited, recent research has been focusing on two dimensions of data acquisition: acquiring redundant passages to support existing questions and acquiring supporting data for answering new questions.

Along the first dimension, gathering redundant passages is likely to boost the confidence of correct answers: Dumais et all [31] make use of the high redundancy of relevant documents on the web and retrieve passages presumed to contain a correct answer. This work supports the intuitive argument that more relevant passages entail higher QA performance. The approach is based on the assumptions that most questions have answers on the web and that the most frequent answer is the correct answer. However, it is less appropriate for questions with sparse supporting web data, multiple meanings, or based on subjective assertions. Extraction models can be learned directly from web data [105] by acquiring very simple but highly redundant patterns or features. Although this approach saturates fast, it is very useful towards answering simple questions.

The second dimension consists of acquiring data to support answering new questions. Girju et all [38] propose a supervised algorithm for part-whole relations based on $20,000$ manually inspected sentences and on $53,944$ manually annotated relations. They report an $F1$ measure of about 90 in answering questions based on part-whole relations. Fleischman et all [33] also propose a supervised algorithm that uses part of speech patterns and a large corpus. The algorithm extracts semantic relations for *Who-is* type questions and builds an offline question-answer database.

Manual acquisition of question-answer pairs is very expensive and highly subjective. However, it is necessary to obtain large amounts of training data for data-driven methods. To overcome this problem, we have proposed a semi-supervised algorithm [70] for high precision question-answer data acquisition from local corpora. The algorithm requires a very small seed data and is compatible with existing question ontologies. More specifically, it makes no assumptions about question types or question structure. Furthermore, our algo-

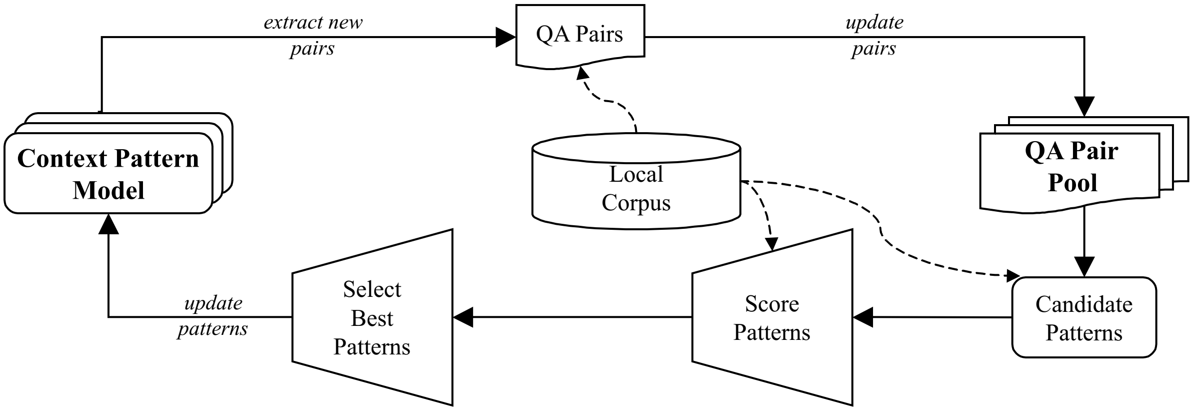

Figure 11.1: Semi-supervised QA data acquisition. During each iteration, question-answer pairs of the same type are used to extract highly correlated context patterns. In turn, the patterns are used to generate more question-answer pairs.

rithm is resource independent and does not assume the availability of specific pre-processing tools.

The approach is also language independent and can be applied to any corpus given an appropriate seed/starting point. We acquired a large set of temporal questions and answers, and we demonstrated their quality through task-based evaluation on TREC question sets.

11.1 Semi-Supervised Data Acquisition Approach

In this work we view questions as collections of entities and relations among them. The missing piece of information – the required answer – is usually in the form of an unknown relation or an unknown entity. Consider the following examples:

A *invented* **Q**

A *is a part of* **Q**

Q *is in* **A**

These statements contain the entities **Q** and **A**, as well as the semantic relations between them. In contrast, actual questions usually consist of incomplete collections of relations and entities. The answering process involves finding the missing element. Some questions may contain all the entities but lack the relation itself:

- What is the connection between **A** and **Q**?
- How are **A** and **Q** related?

while other questions might contain the relation and lack one of the entities involved:

- Who invented **Q**?
- What does **Q** contain?
- Where is **Q**?

where **Q** denotes the entity present in the question and **A** denotes the required answer. We will focus on questions whose answers are missing entities. Relations will also be referred to as **question types** (e.g. *who-invented*, *where-is*), since they usually determine specific answer seeking strategies in most question answering systems.

QA ontologies often include question types as well as answer types. We stress the distinction between answer type and question type: different question types (e.g. *who-invented*, *who-is-the-leader-of*, *who-controls*) may produce answers of the same type (e.g. *person*). For simplicity many existing ontologies often consider question types as specializations of answer types: *leader-of* would be a specialization or refinement of answer type *person*.

The semi-supervised acquisition approach presented here is independent of specific ontologies since we adopt the view that a question type can be directly described through the data: question-answer pairs. For each question type, question-answer pairs (**Q**,**A**) that fit the relation are acquired from the local corpus. Given enough high-quality question-answer pairs, a QA system can be trained to answer similar questions.

Many question answering systems use questions and answers as training data in order to construct or improve their answer seeking strategies. We focus on the process of acquiring high quality question-answer pairs rather than arguing how to incorporate them into a specific QA system.

11.1.1 The Semi-Supervised Algorithm

A relation can be defined as a set of high precision context patterns. The patterns are in fact alternate forms of expressing a concept – for example the relation *who-hired* may occur in raw text as “Y *was hired by* X”, “Y *is employed by* X” etc

The same relation can also be defined indirectly through a set of entity pairs. Each pair is an instance of that particular relation. Sample instances of relation *who-wrote* are: (*Hesse, The Glass Bead Game*) and (*Jefferson, The Declaration of Independence*). Since the relations correspond to question types, the entity pairs can be viewed as question-answer pairs:

- Who wrote “The Glass Bead Game”?
- Who wrote the Declaration of Independence?

We present an semi-supervised algorithm (figure 11.1) that iterates trough these two alternate views of relations: a set of patterns (the *Context Pattern Model*) and a set of question-answer pairs (the *QA Pair Pool*). The algorithm acquires question-answer pairs while at the same time improving the high precision pattern set.

1. Start with a seed of context patterns $\{T\}$ or question-answer pairs $\{(Q, A)\}$
2. Apply the context patterns $\{T\}$ and extract question-answer pairs $\{(Q, A)\}'$ from the local corpus
3. Using the local corpus, extract a set of candidate context patterns $\{T\}'$ that co-occur with $\{(Q, A)\}'$

4. Score the candidate contexts $\{T\}'$ according to a conservative relevance criterion.

5. Select the top K candidate contexts $\{T\}''$

6. Update the model $\{T\}$ with selected contexts $\{T\}''$

7. Return to step 1

11.1.2 Selection Criterion

Each iteration, the Context Pattern Model is updated to contain a subset of the candidate context patterns that have the best scores. Scoring must be based on a criterion that maximizes the correlation of a pattern with the existing question-answer instances in the QA Pair Pool.

The selection criterion used here is the **F1** measure of a pattern T at iteration i. For clarity, we consider the precision and recall of pattern T –which can be thought of as as a query in the local corpus – relative to the known "correct" pair set, QA Pair Pool. Given the QA Pair Pool $\{(Q,A)\}$ during the i^{th} iteration, a candidate context pattern T has a precision and recall:

$$R(T,i) = \frac{PoolCoverage(T,i)}{|Pool(i)|}$$
$$P(T,i) = \frac{PoolCoverage(T,i)}{CorpusCoverage(T)}$$

where $PoolCoverage(T,i)$ is the number of pairs known to be "correct" (i.e. extracted so far and stored in the QA Pair Pool) that were extracted using pattern T as a query in the corpus at iteration i. $CorpusCov(T)$ represents the number of distinct pairs that pattern T can extract from the corpus at iteration i, and $|Pool(i)|$ is the size of the QA Pair Pool at iteration i

The **F1** measure based on pool coverage and corpus coverage is:

$$score_{F1}(T, i) = \frac{2 \cdot P(T, i) \cdot R(T, i)}{P(T, i) + R(T, i)}$$

At iteration $i + 1$, we select the K candidate patterns with highest top F1 score and use them to update the Context Pattern Model.

In order to intuitively illustrate corpus coverage and pool coverage, consider the question type *who-invented*. The goal of the semi-supervised algorithm is to extract as many pairs of inventors and objects invented as possible. The algorithm is considering whether to include the pattern "*A, father of Q*" into the Context Pattern Model. The pattern can be used to extract relevant pairs such as *(Farnsworth, television)*, but also noisy pairs such as *(Michael, John)*. The recall is high is high since many inventors are referred to as parents of their own inventions and consequently this pattern can extract many known pairs inventors-inventions from the corpus: i.e. pairs already in the QA Pair Pool have a high correlation with this pattern. However, the precision is low since the pattern occurs very frequently in the local corpus. As shown in this example, the pattern "*A, father of Q*" is often a manifestation of other relations beside *who-invented*. The corpus coverage of our pattern is high, but only a very small percentage of pair instances actually refer to the inventor-invention relation.

We have explored other selection criteria based on *pool coverage*. These criteria are faster to compute, but very often the algorithm diverges quickly from the original question type. One particular criterion that yields results similar to the F1 measure has been successfully used in semantic lexicon extraction [117]:

$$\begin{aligned} score_p(T, i) &= \frac{PoolCoverage(T, i)}{CorpusCoverage(T, i)} \\ &\quad \cdot \log PoolCoverage(T, i) \end{aligned}$$

Intuitively, pattern T obtains a high score if a high percentage of the pairs it extracts are already in the QA Pair Pool, or if it extracts a moderate number of pairs already in the QA

Pair Pool and it extracts lots of them.

11.1.3 Starting and Stopping Criteria

The algorithm can be initialized either with a small set of patterns in the Context Pattern Model or a set of questions-answer pairs of the same type in the QA Pair Pool. The former approach can be better controlled and has the potential of being more precise, while the later approach can be automated more easily.

A moderate-size validation dataset could be used as the stopping criterion for the algorithm, determining when the question-answer pairs are becoming detrimental as training data to a QA system. When the question-answer pairs extracted are completely deviating from the original relation expressed in the seed, they will most likely not improve the performance of a question answering system, since there is nothing new to be learned. The advantage of a validation set is that the acquisition of question-answer pairs based on different relations will have flexible stopping criteria and the process can be tailored for specific QA systems, rather than imposing a threshold on learning saturation. The disadvantage consists in the fact that standard QA datasets contain very few questions and cover a limited number of question types.

Since using a reasonable-size validation set is not yet feasible, a set of parameters in the semi-supervised algorithm can be learned in order to control how much questions deviate from the original relation. The set of parameters can consist of number of iterations, number of extracted pairs, or a threshold on pattern extraction precision.

11.2 Semantic Drift

Often times questions are either ambiguous or are formulated awkwardly. For example, the question “*Who invented The Muppets?*” is conceptually equivalent to the question “*Who is the creator of The Muppets?*”. The latter formulation is more frequently observed than the former when expressing the connection between Jim Henson and The Muppets. Intuitively, this shows that multiple relations may belong to a larger semantic class.

Semi-supervised algorithms generally experience a degradation in the quality of the data/model over time. Traditionally this is viewed as a negative phenomenon, since it introduces noise. This degradation also varies with the seed data and the corpus being used and is not easily controlled. This phenomenon also occurs in the semi-supervised question-answer pair acquisition algorithm. In practice, conservatively incorporating this noise into the answer extraction model increases the performance.

The very nature of the algorithm dictates that new context patterns will enhance the model after each iteration. We tend to think of these patterns as semantically equivalent. However, in time they tend to diverge from the original relation. We will refer to this semi-supervised algorithm inherent property as **semantic drift**. This property reflects a tradeoff between enriching the original semantic relation and noise in the acquired question-answer pairs.

In our previous example, the answer model starts with the notion of *invention*, accumulating context patterns and question-answer pairs that support the original relation. However, through several iterations, the following context patterns are noticed[1]:

$$
\begin{array}{l}
< inventor\ of > \longrightarrow \\
\quad < creator\ of > \longrightarrow \\
\qquad\qquad < producer\ of > \longrightarrow \\
\qquad\qquad\qquad\qquad < father\ of > \longrightarrow \\
\qquad\qquad\qquad\qquad\qquad\qquad < maker\ of >
\end{array}
$$

While the notions of *creator-of* and *producer-of* could be considered similar to the original relation (*inventor-of*), the subsequent divergence is too generic to produce relevant question-answer pairs.

Similarly, the relation *winner-of* drifts into context patterns referring to people who are

[1] we ignore similar intermediate patterns such as <*the person who invented* > for the purpose of clarity

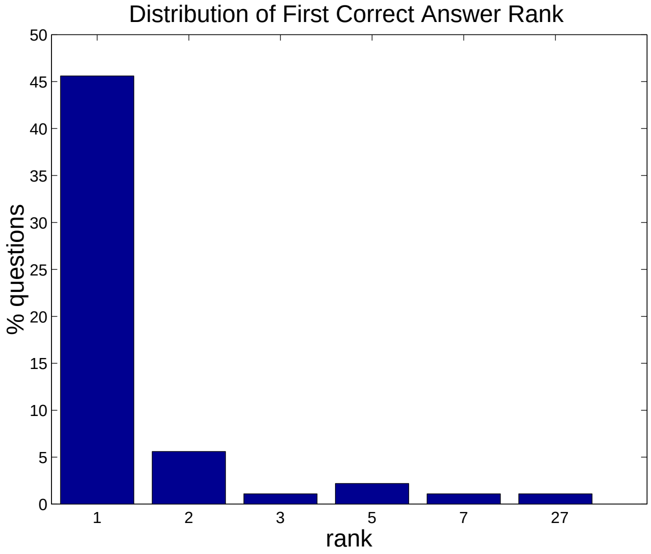

Figure 11.2: High precision data acquisition: most correct answers proposed by the QPairs answer extractor have rank one.

expected to win and have not won yet, while the relation *writer-of* drifts to include patterns about publishers, editors and literary works: (i.e. *A , whose novel Q*).

Ideally, semantic drift can be used to add slightly divergent question-answer pairs to the question pool. However, a critical aspect is the stopping criterion, necessary for deciding when data becomes too noisy to be added to the model. As previously mentioned, a moderate-size validation dataset or a set of learned parameters correlated with noise can be used as the stopping criterion for the algorithm, finding a balance between semantic drift and noise.

With the acquired question answer pairs we used a very simple pattern-based answer extraction method, which is a simplified version of the method described in section 8.2.2. In this chapter we specify implementation details and parameters specific to data acquisition. The answer extractor did not use any question analysis and did not benefit from question term semantic expansion. Also, no answer clustering/merging was performed on the resulting answers. This minimal question answering system was used to test the usefulness of the acquired question-answer pairs and make results reproducible. This way, the results are independent of the particular query expansion, question analysis, or feature implementation.

The retrieval step is trained using the high-precision patterns acquired at each iterations during the esmi-supervised learning. The patterns are added as phrases to simple, keyword-based queries in order to capture more relevant documents. When a new question is processed, several queries are formed by concatenating the question terms and the high-precision patterns. These queries are then used to retrieve the top fifty relevant documents.

We did not want to limit the actual answer extraction to the high precision patterns discovered during the semi-supervised learning. Although very precise, the recall of this set of patterns would have been too low. Therefore, the answer extraction step is trained by extracting a large number of surface patterns (over 5,000) from the local corpus using the question-answer pairs. These patterns range from highly correlated to the question type to weakly correlated. The patterns are further generalized through regular expressions using eliptical terms.

Each pattern's $F1$ score was computed against the question-answer pairs extracted from the local corpora. When a new question is processed, all generalized patterns are applied to the raw documents. Among the ones that do match, the highest scoring patterns are used to extract the final answer set. A more complex answer clustering and merging method is likely to increase QA performance.

The experiments were evaluated using the following metrics:

1. Mean Reciprocal Rank (MRR) - the average reciprocal rank of the first correct answer for each question. Only the top 5 answers are considered for each question.

$$MRR = \frac{1}{N} \cdot \sum_{i=1}^{N} 1/correct\ answer\ rank_i$$

2. Confidence Weighted Score (CWS), which is a measure combining the percent of correct answers and the confidence of the system in its scoring method. Questions are ordered according to confidence in their corresponding answers.

$$CWS = \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{\#\ correct\ up\ to\ question\ i}{i}$$

The context patterns were limited to a maximum size of a sentence. The starting data for the semi-supervised algorithm consists of only one context pattern for each relation:

$$\ldots \quad A \quad <, \, who \, verb > \quad Q \quad \ldots$$

where A and Q are placeholders for the answer and question terms and $verb$ is the verb used to generate the question type. The seed data is extremely simple, but powerful enough that it avoids the human effort that could be put in creating complex and highly precise seeds for each relation. Note that although the seed pattern imposes an ordering on A and Q, the semi-supervised algorithm is free from such constraints.

The learned patterns identify exact answers (i.e. proper names). Text snippets which do not have the correct extent – as defined by answer patterns provided by NIST – are considered incorrect answers. The algorithm was run for each relation, producing up to 2,000 question-answer pairs per question type. For more obscure relations such as *who-found*, the algorithm acquired fewer pairs than for more common relations such as *who-made*.

Using this bare-bones question answering system, we obtained an overall MRR score for TREC test data of 0.54 with the confidence weighted score CWS if 0.73. Figure 11.2 shows the overall rank distribution of first correct answers. On the same temporal data, the top five performing systems at TREC obtained scores ranging between 0.4 MRR and 0.76 MRR.

Figure 11.3 compares the performance of our answer extraction engine (referred to as *QAPairs*) with the performance of the top five systems at TREC 9, 10, and 11 on the same test data. Note that different systems obtained the top five results in different years. The results are significant, especially when taking into account that the top five systems are full-fledged QA systems incorporating knowledge resources, specialized document retrieval, complex question and passage processing, answer selection and verification. In contrast we focused on a simple answer extraction component of a question answering system in order to show the high potential of using additional question-answer pairs in training QA systems. Although data acquisition is not the purpose of this dissertation, a possible extension to an instance-based QA system would be a system-driven data acquisition. More specifically, for every cluster, a set of new similar questions and corresponding answers could be acquired. The

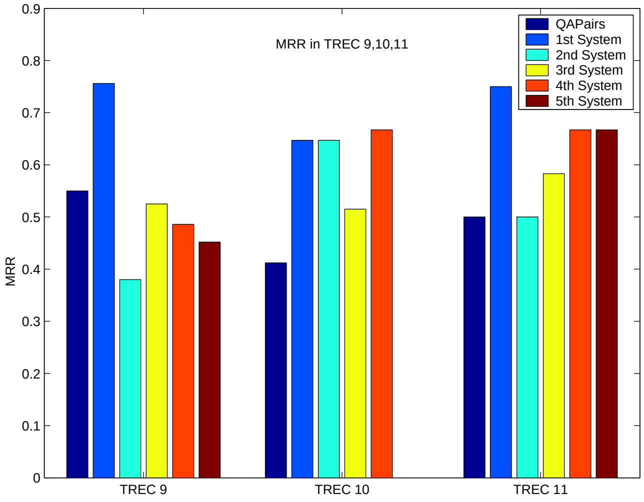

Figure 11.3: QAPairs compared to the top five system performance at TREC 9, 10 and 11 on the *same* test data used in our experiments.

quality of these questions and answers could be tested on the original training data, before being incorporated into the question answering system. This would help improve the IBQA performance using a more conservative QA data acquisition.

With each iteration, the semi-supervised algorithm acquires more question-answer pairs. At each iteration the answer extractor is re-trained and evaluated. Figure 11.4 shows how performance improves with each iteration. Advanced iterations contribute to the QA process by answering more ambiguous questions and capturing answers which are awkwardly stated. However, as more question-answer pairs are added to the pool, they become more obscure and contribute less to learning new patterns.

The fact that performance increases with the acquisition of more question-answer pairs shows that the scoring method correlates well with the number of iterations. The more training data is obtained from the local corpus, the better the answer extraction component performs. This observation further suggests that more complex question answering systems can take better advantage of the acquired data.

The semi-supervised algorithm is easy to implement and adapt to specific question an-

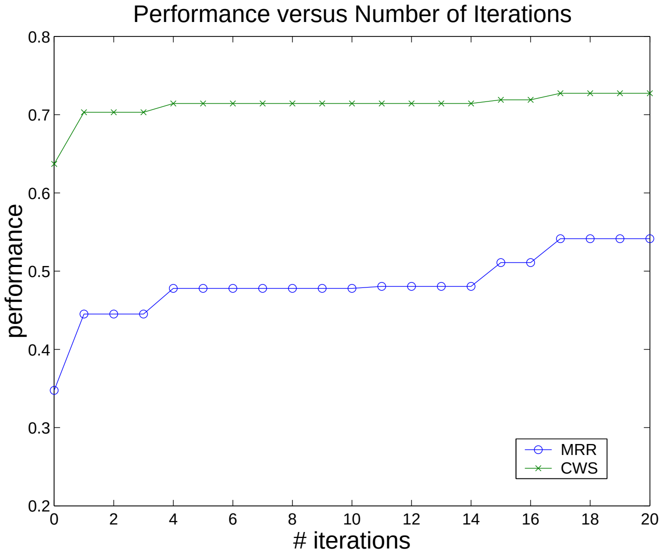

Figure 11.4: Performance increases with the number of iterations and therefore with the size of the training data.

swering systems. With each iteration, pairs acquired by the semi-supervised algorithm are used as training data to a simple QA system. Performance increases with the number of question-answer pairs acquired confirming the robustness of the semi-supervised algorithm.

Current work in question answering data acquisition suggests the availability of larger question-answer datasets in the near future, thus enabling statistical, data-driven techniques to become more feasible. Larger question-answer datasets would also support the development of more approaches in the spirit of instance-based question answering which relies solely on questions and corresponding correct answers as training data.

11.2.1 Qualitative Analysis

Table 11.1 shows qualitative results produced by the semi-supervised algorithm. Five sample relations are presented with question-answer pair sampling at 1, 10, 100, and 1000 as more data was added to the pool. The specificity varies from very exact questions pairs such as "*Who owns the New Jersey Devils?*" to broader questions more likely to have many correct

Pair#	**Question Term**	**Answer**
who-invented		
1	dynamite	Alfred Nobel
10	theosophy	Helena Blavatsky
100	dialectical philosophy	Hegel
1,000	television's Twin Peaks	Mark Frost
who-created		
1	Providence's Waterfire	Barnaby Evans
10	Howdy Doody	Buffalo Bob Smith
100	HBO's acclaimed Mr. Show	Troy Miller
1,000	the invisible holster	Charlie Parrot
who-makes		
1	small motors	Johnson Electric Holdings
10	ping golf clubs	Karsten Manufacturing corp.
100	removable media data storage devices	Iomega corp.
1,000	all the airbus wings	British Aerospace
who-owns		
1	The Candlelight Wedding Chapel	Gordon Gust
10	The New Jersey Devils	John Mcmullen
100	the sky diving operation	Steve Stewart
1,000	the ambulance company	Steve Zakheim
who-founded		
1	Associated Publishers inc.	Mr. Cox
10	Earthlink Network	Sky Dayton
100	Limp Bizkit's label	Jordan Schur
1,000	Macromedia	Marc Canter

Table 11.1: Sample qualitative results. Question-answer pairs are added to the pool incrementally. We show the 1^{st}, 10^{th}, 100^{th}, $1,000^{th}$ question-answer pairs as they are added to the pool.

answers - i.e. "*Who makes small motors?*". In order to show the semantic similarity between two question types as seen through the data, we included both the *invented* and *created* relations.

CHAPTER 12

IBQA Conclusions & Future Work

In this dissertation, we have introduced IBQA, a fully statistical, data-driven, instance-based approach to question answering in which we learn how to answer new questions from similar training questions and their known correct answers. Under this approach, training questions are clustered based on different similarity metrics. We automatically learn answering strategies for answering questions belonging to individual clusters. Several answering strategies are simultaneously employed, based on the clusters the new questions falls under. Each cluster-specific answering strategy consists of an expected answer model, a query content model, and an answer extraction model. We apply these models successively to analyze the question, retrieve relevant documents and extract correct answers.

The core instance-based approach does not rely on resources such as: WordNet, parsers, taggers, ontology, hand-coded optimizations, and hand-coded patterns. However, we show that our approach can easily integrate and benefit from resources such as a morphological analyzer and WordNet, used for for synonymy and semantic classes for answer types. The

IBQA approach is resource-friendly, allowing external resources to be incorporated. To further specialize an instance-based system to specific domains, rule-based components may also be used at every stage in the question answering process – e.g. rule-based answer extractors, answer type extractors, or rule-based query content generation.

The main component of the instance-based question answering approach are data-driven. Rather than applying an expert-designed pre-defined answering strategy to new questions, we automatically learn cluster-specific strategies, estimate the probability of success for each of the strategy models, and we directly incorporate the estimates into the overall answer score. The IBQA approach allows training question datasets to drive the question clustering. The dataset composition also determines how accurate answering strategies are when they are learned from their corresponding clusters. Document corpora from which we retrieve the raw documents also directly influence what models can be learned and what questions can be successfully answered by these models.

In the document retrieval stage of IBQA, from each cluster of training questions we automatically derive additional specialized query content in order to focus and enhance queries, and consequently improve the likelihood of success of retrieval in the QA process. In section 7.2 we show the additive benefit of several more traditional query expansion methods as well as the impact of our cluster-based expansion. We show that queries expanded using our cluster-based method can retrieve new, relevant documents that otherwise couldn't be retrieved using standard expansion. This provides answer extraction with a more diverse set of relevant documents, potentially improving extraction – depending on the answer extraction method used.

We have shown that for factoid as well as for definitional questions, the instance-based approach provides a very high baseline, without relying on extensive resources, processing tools or human expertise. Since training questions guide the answering strategy learning process, the instance-based approach can be extended to more than factoid questions. We show that without tailoring our IBQA system, we obtain good results for person-profile and object definition questions. A possible improvement to the IBQA approach that may better equip it to deal with definitional questions consist of shallow parsing. While less specific than full parsing, shallow parsing is more robust and obtains better performance. In question answering, this is an important concern since errors obtained while processing questions and documents accumulate and propagate throughout the QA process.

A promising future work direction for IBQA consists of further investigating the issues involved in applying our approach to new languages. While the core instance-based question answering approach is language independent and can easily be re-trained for individual languages, in order to obtain a good performance, language-specific pre-processing might be required (e.g. different encodings, different grammar, different document distributions and corpus density etc). The IBQA approach does not depend on language-specific resources or manual parameter optimization, but it allows the integration of language-dependent tools: part of speech tagging, parsing, and named entity tagging.

Since the instance-based QA is fully trainable and does not rely on hand-written rules and hand-tuned parameters, it allows for fast re-training, with little human effort. For example given the processed TREC training data questions and corresponding answers, an IBQA system can be trained in several days to several weeks, depending on the number of clusters, feature set size, number and type of answer extractors used, availability and access speed of the retrieval engine. After training, the IBQA system is a good platform for further question answering research. Further performance improvements could be obtained by incorporating additional resources (e.g. ontologies, gazetteers) and processing tools (e.g. part of speech tagging, shallow parsing) as well as by increasing the training data size – i.e. adding more training questions, correct answers, and retrieving more relevant documents.

12.1 Strategy Selection

An increasing number of question answering systems are relying on multi-strategy approaches in order to find answers to questions. They rely on multiple question classifications, answer extractors, multiple retrieval methods using several data sources, and different web-based services. While question answering performance is often presented on batch processing of questions with no time constraints, in real-life scenarios, only a limited number of these strategies can be fully explored. Under these scenarios response time and performance trade-offs require careful selection of answering strategies such that performance is optimized subject to constraints.

In the instance-based question answering approach, depending on the clustering algorithms, the size and distribution of the training dataset, a QA system that fully explores all available strategies can be very slow. We are interested whether selecting a small number of strategies according to confidence scores could result in a limited overall performance degradation, while considerably reducing the answering strategies utilized.

In this dissertation we have presented a strategy selection approach that directly addresses these issues and we apply it to a statistical instance-based question answering system. Through experiments we have shown the significant benefits of a principled strategy selection method on document retrieval, answer extraction, and answer merging (i.e. overall QA performance) using several metrics. By carefully selecting 10% of the available answering strategies, we show that an instance-based question answering system can obtain similar performance to the scenario in which we employ all strategies. Moreover, the cluster-based confidence scoring method was also incorporated into answer merging which improved performance both in terms of MRR and Top5 significantly.

12.2 Extensibility of a Data-Driven QA Approach

The instance-based approach to question answering relies on the availability of datasets of training questions and corresponding answers. More training data entails better coverage for new test questions and better cluster models. In recent years the acquisition of such datasets has become an active research direction. In section 11 we show that research on large scale data acquisition for question answering is advancing and promises to produce large datasets of questions and answers. We presented a semi-supervised algorithm [70] for high precision question-answer pair acquisition from local corpora, that is able to acquire high quality training data using very small seeds. Based on the newly acquired question answering pairs, we trained a bare-bones QA system, including a very simple, pattern-based answer extractor. This bare-bones system obtained good MRR and CWS performance on TREC test data. On the same test data, the performance of this system is comparable to the fifth performing system at TREC.

Many question answering systems employ various resources such as WordNet, gazetteers, encyclopedias, and dictionaries. Processing tools such as morphological analyzers, part of speech taggers, and parsers are also widely used to improve the performance of question answering systems. If data-driven QA systems can seamlessly make use of these resources and tools, they can benefit from the human knowledge incorporated in these specialized components. The performance of a baseline instance-based QA system can be improved by further incorporating such resources. It offers a principled, robust, and reproducible platform for evaluating the impact of various resources and processing tools.

12.3 Future Work

A central problem in question answering is the lack of standardization. This makes it very difficult to compare QA systems solely on the overall results they report, usually in the form of an overall number describing the overall system performance. Systems use different resources, different pipeline stages, different retrieval, extraction, and merging models etc. Moreover, the overall results of of QA pipeline can sometimes overshadow high quality components or hide the performance of problematic components. For example a great answer extraction model cannot extract correct answers given a low quality retrieval component, and vice-versa. Partly because of these issues, most existing question answering systems cannot be fully re-implemented based on available publications and documentation and their results cannot be fully reproduced.

A first step towards solving this problem consists of open, language independent QA frameworks with more standardized interfaces between components. Within such frameworks, individual components can more easily be tested for performance, robustness, and efficiency. Another potential solution is for official evaluations to periodically introduce question datasets from *surprise* domains and languages, requiring QA systems to be more adaptable. This will also move question answering research towards more open QA systems and more easily reproducible experiments.

Advances in statistical, data-driven question answering systems open the door for new and exciting research directions and offer a fresh view on existing QA-related problems and approaches. Current pipeline approaches can be adapted to handle more data and systems can be trained faster, using less human involvement. Question answering systems may be more easily ported to new types of questions, domains, and languages.

Document and passage retrieval can shift from the more static role in QA to a dynamic, trainable component, where the types of queries, as well as query content are learned from training data. In particular, a natural extension to IBQA is learning structured queries and investigating methods for automatically incorporate semantic and resources and syntax in retrieval, in order to increase the number of relevant documents that include correct answers in easy to extract contexts. Perhaps more importantly, some question types (or cluster-specific strategies) may benefit from certain resources, while others may not. A difficult challenge is to automatically select the level of contribution for these resources. For example: what WordNet synsets should be used in query expansion and should synsets be weighted depending on their ranks and relevance; how should hypernyms be included in a structured query, and how many hypernyms should be considered; what is the relative importance of proper nouns compared to common nouns, and what role should shallow parsing have in defining phrase boundaries for retrieval

Answer extraction under IBQA allows the potential for several extraction models to be built and tested for every cluster-based strategy. This would have the effect of making answering strategies more flexible and potentially adapt better to each cluster and documents retrieved. Since different extractors have different biases, they may be more suited for different types of answers or contexts. This approach would increase the complexity of the answer merging component since the same answer could be extracted by multiple extractors, making the problem of estimating answer correctness, as well as merging it with other similar answers, more complex.

In terms of scalability, new selection methods may perform better, and even have the potential of outperforming the greedy oracle selection strategy. In particular, active learning approaches may allow for faster training when applied to strategy selection. In particular, we are interested in the effect of incremental training data on the selection of answering strategies.

Since our approach is trainable using available questions and answers, a natural research direction is to apply the instance-based approach to new domains. Technical domains often have specific question formats and specific answer structures, and trainable question answering systems have the advantage of quick portability from the *open-domain* (i.e. trivia questions and news documents) to fields such as medical, biology, anthropology, which provide different test beds for a question answering system.

Complex questions are very often tackled by breaking them up into simpler (often factoid) supporting questions and answering them first. Answering complex questions such as: "*What successful products made by Microsoft's partners have been recently advertised in India?*" may require first answering questions such as "*Who are Microsoft's partners?*", "*What products does company X make?*", and "*What products have been recently advertised in India?*" or '*Has product X been recently advertised in India?*". An instance-based QA system can be utilized as a factoid question answering component and has the advantage of providing individual component success estimates that can be incorporated by an overseeing planner or reasoning component. FAQ questions, are a more open-ended type of complex questions. For FAQ questions, the expected answer is defined to be a paragraph rather than a phrase and the function to be optimized is an overlap-based measure (such as Rouge) rather than MRR or TREC score.

Another future work direction is the application of IBQA to languages other than English. A mono-lingual instance-based question answering system would be trained in the same manner as the English. However, different pre-processing of questions and documents (such as feature extraction, sentence splitting, part-of-speech tagging and morphological analysis) would be required. Particularly difficult is identifying the components of an IBQA system that are language independent and decoupling them from the rest of the system, to make porting to other languages easier. Another extension to the IBQA approach is modifying it for cross lingual question answering, where questions are provided in a source language and answers have to be extracted in a different, target language. It is not sufficient to translate the questions and subsequently perform monolingual QA since translation quality varies, and keyword expansion is more difficult and depends on context and on the particular language-pairs. Furthermore question surface-form patterns are not as reliable as in monolingual question answering, and generating retrieval queries must take into account more complex representation of the translated question.

12.4 Towards Applying IBQA to New Languages and Domains

A major advantage of the instance-based approach to question answering is the fact that it is a fully statistical approach, easily trainable using new datasets. This section explores the issues involved in modifying and re-training an IBQA system for new languages or domains. The first issue that needs to be addressed deals with the raw **training data**. Depending on the domain/language, training data may not be very easy to obtain or generated. The training question distribution should be as similar as possible to the expected test question distribution. Moreover, for all available training questions, it is necessary to ensure that answer keys cover the top most frequent answers and answer forms in the local corpus or the web. This will improve the query content model and the extraction training data, better differentiating the positive examples from the negative ones.

In terms of **question processing**, the most basic features to be used as dimensions are the actual lexical items, or more generally n-grams based on the lexical items. Depending on the language, processing tools such as part-of-speech taggers, parsers, or shallow parsers can be used if available and if their performance is reasonable. Typically, such tools are designed for non-interrogative text and may have to be adapted to work well with questions. For languages that use capitalization (e.g. not Chinese or Arabic) for named entities, IBQA can make use of NE classing (e.g. “Bob Marley” as a ¡ProperNounPhrase¿). Furthermore several languages use special conventions for marking titles or acronyms. These can be also marked as special classes and used as features for dimensions in the question vector space.

Another concern when porting an IBQA system to another dataset is **answer type** representation. For popular culture and news stories, WordNet covers a sufficeint number of answer types. For a different domain, a new set of answer types may be requried. For example, for the medical domain, a medical dictionary or MESH (Medical Subject Headings) could be used as the set of expected answer types. Several WordNets have been developed for languages other than English and these can be used if IBQA is ported to these languages. For languages with limited resources, a small set of answer types or a small answer type ontology based on training question data can be created.

The **document retrieval** step can be easily adapted to new domains or languages. Depending on the corpus and the retrieval engine used, a practical issue is modifying the query structure and the interface to the search engine. More importantly, the keywords extracted from the query are expanded – in the case of TREC questions, English language and web corpus: through synonym expansion using WordNet and inflectional transformations using a morphological analyzer. Similar processing could be used when porting IBQA but using different resources. For languages such as Chinese, for the purpose of retrieval, plural/singular or tense detection (from context) are not required. Several parameters can be modified depending on the particular training question set: the number of documents retrieved, the feature selection method, the type and usefuleness of query expansion to be employed, and the size of the query content model, which translates into the number of expanded queries needed to improve retrieval.

For the **answer extraction** component, depending on the answer type structure, relevant document/passage density and context structure typical for the specific domain, different extractors will have different performance. The same methods can be used as presented in this document (and implemented in our system), but several experiments to test the minimum data requirements should be performed - e.g. number of positive vs. negative training sentences: correct vs. incorrect answers. As a backoff strategy, the proximity-based extractor provides a good baseline. However, the pattern-based extractor and the SVM-based extractor may or may not be the most appropriate extraction methods.

Simple **answer merging** is very similar across languages and domains for IBQA, since we automatically compute answer confidence based on cluster quality and extraction scores. However, for more complex answer merging, where further processing is required or where partially overlapping answers are involved, language-dependent processing and merging methods could be required. These modifications would be similar in techical domains where answer structure is more complex: i.e. chemical formulas or Linux commands.

From a practical perspective, in the overall document procssing, character encoding issues must first be resolved. For languages that make use of diacritics, the use of an approximation mapping should be investigated since in many cases authors omit various marks. This happens especially in lower quality text such as web-documents as opposed to news stories. If any constraints are used in question clustering, the corresponding parameters have to be re-considered. For example, for news stories and TREC questions, clusters with at least three questions can be useful. Therefore, we have used a constraint of a minimum of three training questions per clusters. However, different domains and different document densities in available corpora may require a stricter constraint.

Acknowledgments

First and foremost I would like to thank my advisor, Jaime Carbonell and my committee members: Eric Nyberg, Tom Mitchell, and Nanda Kambhatla, whose advice, support, and invaluable feedback allowed me to finish this dissertation.

At CMU I had the opportunity to have Jaime Carbonell as my Ph.D. advisor. He supported my research in the NLP realm, allowed me to actively explore many different problems and go where my research interests took me. I have worked closely with Eric Nyberg on the Javelin project, learning a great deal from him as a mentor and a project leader. I was fortunate to have Sebastian Thrun as my first year mentor and advisor. His research guidance and support gave me my bearings in grad school.

On the CMU Javelin project, I had the opportunity to work with professors Teruko Mitamura, Bob Frederking, and Jaime Callan. Off the well-traveled road, prof. Alon Lavie supported my research in machine translation evaluation and actively encouraged me to pursue my ideas. At CMU I actively collaborated and/or carried inspiring discussions with: Monica Rogati, Yiming Yang, Roni Rosenfeld, Danny Slater, Laurie Hiyakumoto, Krzysztof Czuba, Paul Bennett, John Langford and many others.

I am grateful for awesome friends who gave me have a life outside my Ph.D. work:

Stefan, Jernej, Vahe, Vince, Christina, Mihnea, Veronica, Bogdan, Leo, Mugizi, Tadashi, Tiankai, Rebecca and others.

Salim Roukos and my internships at IBM TJ Watson have had the single most significant impact on shaping my research thinking. At IBM I was fortunate to have excellent mentors and extremely bright colleagues: Nanda Kambhatla, who advised me both as an internship mentor and as a thesis committee member, Kishore Papineni, who sparked my interest in machine translation, Abe Ittycheriah, whose friendship and insight I deeply appreciate, Todd Ward, Yaser Al-Onaizan, Radu Florian, Nyu Ge, Nicolas Nicolov, Dan Bikel, XiaoQiang Luo and many others. Most importantly, during my work at IBM, I learned from Salim Roukos how to ask the right questions and how to perform thorough, meaningful research.

My interest in research and graduate school has been shaped by my UC Davis undergraduate mentors: Charles Martel, Michael Gertz, Debbie Niemeier, and most of all by my computer architecture research advisor prof. Fred Chong, whose support, trust, and encouragement convinced me to continue on a research path. I was also influenced by the friendship and imaginative research minds of: Mark Oskin, Justin Hensley, and Diana Franklin.

I due my math and computer science background to my early instructors: Bunea, Stoica, Mitrache, Sorin, Cherciu, Mihaileanu, and Hansen, all of whom helped build the foundations needed throughout my Ph.D.

My family (disambiguation: grandparents, parents, brother, in-laws) has been most supportive throughout this endeavor and throughout my life. My grandparents and my parents, but especially my grandmother, who took countless ours to make sure my homeworks were done right and the knowledge absorbed properly. My father and Gazeta Matematica were a particularly tough couple, difficult but fun to defeat, and my mother stimulated my tech imagination with tales of her robots. During grad school Kuki was the best stress reliever anyone could ever have.

Most importantly, throughout my Ph.D. I was immensely fortunate to have my wife Monica's love, support, and extremely sharp technical conversations. She kept my sanity almost intact, my research focused, my life grounded in reality, and my every compass pointing towards North $\pm 1^\circ$.

Thank you.

Bibliography

[1] S. Abney. Partial parsing via finite-state cascades. In *Journal of Natural Language Engineering*, 1996.

[2] S. Abney, M. Collins, and A. Singhal. Answer extraction. In *Conference on Applied Natural Language Processing (ANLP)*, 2000.

[3] E. Agichtein, S. Lawrence, and L. Gravano. Learning search engine specific query transformations for question answering. 2001.

[4] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 1990.

[5] D. Azari, E. Horvitz, S. Dumais, and E. Brill. Web-based question answering: A decision-making perspective. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

[6] P. N. Bennett. Using asymmetric distributions to improve text classifier probability estimates. *ACM Special Interest Group on Information Retrieval Conference (SIGIR)*, 2003.

[7] P. N. Bennett. Classifier probability recalibration toolkit v1.0. *Carnegie Mellon University*, 2005.

[8] D. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: A high-performance learning name finder. 1997.

[9] D. Bikel, R. Schwartz, and R. Weischedel. An algorithm that learns what's in a name. In *Machine Learning*, 1999.

[10] M. W. Bilotti, B. Katz, and J. Lin. What works better for question answering: Stemming or morphological query expansion? In *Information Retrieval for Question Answering (IR4QA), SIGIR Workshop,* 2004.

[11] S. Blair-Goldenshon, K. McKeown, and A. Schlaikjer. Defscriber: A hybrid system for definitional qa. *ACM Special Interest Group on Information Retrieval Conference (SIGIR),* 2003.

[12] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. *Annual Conference on Learning Theory (COLT)*, 1992.

[13] E. Breck, J. Burger, L. Ferro, L. Hirschman, D. House, M. Light, and I. Mani. How to evaluate your question answering system every day and still get real work done. In *International Conference On Language Resources And Evaluation (LREC),* 2000.

[14] E. Breck, J. Burger, L.Ferro, D. House, M. Light, and I. Manni. A system called qanda. In *Text REtrieval Conference (TREC-8),* 1999.

[15] E. Brill, S. Dumais, and M. Banko. An analysis of the askmsr question answering system. In *Conference on Empirical Methods in Natural Language Processing (EMNLP),* 2002.

[16] E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data-intensive question answering. In *Text REtrieval Conference (TREC),* 2001.

[17] J. Burger, L. Ferro, W. Greiff, J. Henderson, M. Light, and S. Mardis. Mitre's qanda at trec-11. In *Text REtrieval Conference (TREC),* 2002.

[18] J. Chu-Carroll, K. Czuba, J. Prager, and A. Ittycheriah. In question answering, two heads are better than one. In *Human Language Technology Conference - North American chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2003.

[19] C. Clarke, G. Cormack, G. Kemkes, M. Laszlo, T. Lynam, E. Terra, and P. Tilker. Statistical selection of exact answers. In *Text REtrieval Conference (TREC)*, 2002.

[20] C. Clarke, G. Cormack, D. Kisman, T. Lynam, and E. Terra. Question answering by passage selection (multitext experiments for trec-9. In *Text REtrieval Conference (TREC)*, 2000.

[21] C. Clarke, G. Cormack, and T. Lynam. Exploiting redundancy in question answering. In *ACM Special Interest Group on Information Retrieval Conference (SIGIR)*, 2001.

[22] C. Clarke and E. L. Terra. Passage retrieval vs. document retrieval for factoid question answering. In *ACM Special Interest Group on Information Retrieval Conference (SIGIR)*, 2003.

[23] K. Collins-Thompson, E. Terra, J. Callan, and C. Clarke. The effect of document retrieval quality on factoid question-answering performance. In *ACM Special Interest Group on Information Retrieval Conference (SIGIR)*, 2004.

[24] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[25] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 1995.

[26] W. Croft, S. Cronen-Townsend, and V. Lavrenko. Relevance feedback and personalization: A language modeling perspective. In *DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries*, 2001.

[27] H. Cui, M.-Y. Kan, and T.-S. Chua. Generic soft pattern models for definitional question answering. In *ACM Special Interest Group on Information Retrieval Conference (SIGIR)*, 2005.

[28] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 1977.

[29] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons Inc., 2001.

[30] S. Dudoit and J. Fridlyand. A prediction based resampling method to estimate the number of clusters in a dataset. *Genome Biology*, 2002.

[31] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: Is more always better? In *ACM Special Interest Group on Information Retrieval Conference (SIGIR)*, 2002.

[32] A. Echihabi and D. Marcu. A noisy channel approach to question answering. In *The Association for Computational Linguistics Conference (ACL)*, 2003.

[33] M. Fleischman, E. Hovy, and A. Echihabi. Offline strategies for online question answering: Answering questions before they are asked. In *The Association for Computational Linguistics Conference (ACL)*, 2003.

[34] R. Florian, H. Hassan, A. Ittycheriah, H. Jing., N. Kambhatla, X. Luo, N. Nicolov, and S. Roukos. A statistical model for multilingual entity detection and tracking. In *Human Language Technology Conference - North American chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2004.

[35] J. Fukumoto, T. Kato, and F. Masui. Question answering challenge (qac-1): An evaluation of question answering task at ntcir workshop 3. In *NTCIR Workshop3*, 2002.

[36] I. Gabbay and R. F. Sutcliffe. A qualitative comparison of scientific and journalistic texts from the perspective of extracting definitions. *Workshop on Question Answering In Restricted Domain at ACL*, 2004.

[37] R. Girju. Answer fusion with on-line ontology development. In *North American chapter of the Association for Computational Linguistics (NAACL)*, 2001.

[38] R. Girju, D. I. Moldovan, and A. Badulescu. Learning semantic constraints for the automatic discovery of part-whole relations. In *HLT-NAACL*, 2003.

[39] H. Hacioglu and W. Ward. Question classification with support vector machines and error correcting codes. In *Human Language Technology Conference - North American chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2003.

[40] S. Harabagiu and F. Lacatusu. Strategies for advanced question answering. In *HLT-NAACL Workshop on Pragmatics of Question Answering*, 2004.

[41] S. Harabagiu, S. J. Maiorano, A. Moschitti, and C. A. Bejan. Intentions, implicatures and processing of complex questions. In *Workshop on Pragmatics of Question Answering at HLT-NAACL 2004*, 2004.

[42] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, R. Bunescu, R. Girju, V. Rus, and P. Morarescu. Falcon: Boosting knowledge for answer engines. In *Text REtrieval Conference (TREC)*, 2000.

[43] T. Hastie, R. Tibshirani, and J. Friedman. *The Element of Statistical Learning. Data Mining, Inference, and Prediction*. Springer Series in Statistics, 2001.

[44] U. Hermjakob. Parsing and question classification for question answering. In *ACL Workshop on Open-Domain Question Answering*, 2001.

[45] U. Hermjakob, A. Echihabi, and D. Marcu. Natural language based reformulation resource and web exploitation for question answering. *Text REtrieval Conference (TREC)*, 2002.

[46] U. Hermjakob, E. Hovy, and C. Lin. Knowledge-based question answering. In *Text REtrieval Conference (TREC)*, 2000.

[47] W. Hildebrandt, B. Katz, and J. Lin. Answering definition questions using multiple knowledge sources. In *Human Language Technology Conference - North American chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2004.

[48] L. Hiyakumoto, L. V. Lita, and E. Nyberg. Multi-strategy information extraction for question answering. In *The FLorida Artificial Intelligence Research Society Conference (FLAIRS)*, 2005.

[49] E. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C. Lin. Question answering in webclopedia. In *Text REtrieval Conference (TREC)*, 2000.

[50] E. Hovy, L. Gerber, U. Hermjakob, C. Lin, and D. Ravichandran. Toward semantics-based answer pinpointing. In *Human Language Technology Conference(HLT)*, 2001.

[51] E. Hovy, U. Hermjakob, and C. Lin. The use of external knowledge in factoid qa. In *Text REtrieval Conference (TREC),* 2001.

[52] E. Hovy, U. Hermjakob, C. Lin, and D. Ravichandran. Using knowledge to facilitate factoid answer pinpointing. In *International Conference on Computational Linguistics (COLING),* 2002.

[53] E. Hovy, U. Hermjakob, and D. Ravichandran. A question/answer typology with surface text patterns. In *Human Language Technology Conference(HLT),* 2002.

[54] A. Ibrahim, B. Katz, and J. Lin. Extracting structural paraphrases from aigned monolingual corpora. In *International Workshop on Paraphrasing (IWP): Paraphrase Acquisition and Applications*, 2003.

[55] A. Ittycheriah, M. Franz, and S. Roukos. Ibm's statistical question answering system - trec-10. In *Text REtrieval Conference (TREC)*, 2001.

[56] A. Ittycheriah, M. Franz, and S. Roukos. Ibm's statistical question answering system - trec-11. In *Text REtrieval Conference (TREC)*, 2002.

[57] A. Ittycheriah, M. Franz, W. Zhu, and A. Ratnaparkhi. Ibm's statistical question answering system - trec-9. In *Text REtrieval Conference (TREC)*, 2000.

[58] T. Joachims. *"Learning to Classify Text Using Support Vector Machines*. Kluwer, 2002.

[59] T. Joachims. Learning to classify text using support vector machines. *Ph.D. Dissertation*, 2002.

[60] B. Katz, J. Lin, D. Loreto, W. Hildebrandt, M. Bilotti, S. Felshin, A. Fernandes, G. Marton, and F. Mora. Question answering from the web using knowledge annotation and knowledge mining techniques. In *Text REtrieval Conference (TREC),* 2003.

[61] J. Ko, L. Hiyakumoto, and E. Nyberg. Exploiting multiple semantic resources for answer selection. In *International Conference On Language Resources And Evaluation (LREC),* 2006.

[62] C. C. T. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the web. In *The World Wide Web Conference (WWW)*, 2001.

[63] W. Li, R. Srihari, C. Niu, and X. Li. Entity profile extraction from large corpora. *Pacific Association for Computational Linguistics Conference (PACLING)*, 2003.

[64] X. Li and D. Roth. Learning question classifiers. In *International Conference on Computational Linguistics (COLING)*, 2002.

[65] C.-Y. Lin. Rouge: a package for automatic evaluation of summaries. In *ACL Workshop on Text Summarization*, 2004.

[66] J. Lin and D. Demner-Fushman. Automatically evaluating answers to definition questions. In *Human Language Technology Conference - Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, 2005.

[67] J. Lin and B. Katz. Question answering from the web using knowledge annotation and knowledge mining techniques. In *Conference on Information and Knowledge Management (CIKM)*, 2003.

[68] L. V. Lita, , A. Schlaikjer, W. Hong, and E. Nyberg. Qualitative dimensions in question answering: Extending the definitional qa task. In *American Association for Artificial Intelligence Conference (AAAI)*, 2005.

[69] L. V. Lita and J. Carbonell. Instance-based question answering: A data-driven approach. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2004.

[70] L. V. Lita and J. Carbonell. Unsupervised question answering data acquisition from local corpora. *Conference on Information and Knowledge Management (CIKM)*, 2004.

[71] L. V. Lita and J. Carbonell. Cluster-based selection of statistical answering strategie. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2007.

[72] L. V. Lita, W. Hunt, and E. Nyberg. Resource analysis for question answering. *The Association for Computational Linguistics Conference (ACL)*, 2004.

[73] X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. A mention-synchronous coreference resolution algorithm based on the bell tree. In *ACL*, 2004.

[74] S. Macnaughton. *Dissimilarity Analysis: A New Technique of Hierarchical Analysis*. Univ. of California Press, 1965.

[75] B. Magnini, S. Romagnoli, A. Vallin, J. Herrera, A. Penas, V. Peiado, F. Verdejo, and M. de Rijke. The multiple language question answering track at clef 2003. In *Cross Language Evaluation Forum (CLEF)*, 2003.

[76] B. Magnini, A. Vallin, C. Ayache, G. Erbach, A. Penas, M. de Rijke, P. Rocha, K. Simov, and R. Sutcliffe. Overview of the clef 2004 multilingual question answering track. In *Cross Language Evaluation Forum (CLEF)*, 2004.

[77] G. Mann. A statistical method for short answer extraction. In *The Association for Computational Linguistics Conference (ACL)*, 2001.

[78] G. Mann. Fine-grained proper noun ontologies for question answering. In *Building and Using Semantic Networks (SemaNet) - workshop in conjunction with COLING*, 2002.

[79] G. Mann. Learning how to answer questions using trivia games. In *International Conference on Computational Linguistics (COLING)*, 2002.

[80] G. Marton and A. Radul. Nuggeteer: Automatic nugget-based evaluation using descriptions and judgements. In *Human Language Technology Conference - North American chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2006.

[81] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. *AAAI, Workshop on Learning for Text Categorization*, 1998.

[82] G. A. Miller, R. Beckwith, C. D. Fellbaum, D. Gross, and K. Miller. Five papers on wordnet. *International Journal of Lexicography*, 1990.

[83] G. W. Milligan and M. Cooper. Examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 1985.

[84] G. Minnen, J. Carroll, and D. Pearce. Applied morphological processing of english. *Journal of Natural Language Engineering*, 2001.

[85] D. Moldovan, D. Clark, S. Harabagiu, and S. Maiorano. Cogex: A logic prover for question answering. In *The Association for Computational Linguistics Conference (ACL)*, 2003.

[86] D. Moldovan, S. Harabagiu, R. Girju, P. Morarescu, F. Lacatusu, A. Novischi, A. Badulescu, and O. Bolohan. Lcc tools for question answering. In *Text REtrieval Conference (TREC)*, 2002.

[87] D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Girju, R. Goodrum, and V. Rus. The structure and performance of an open-domain question answering system. In *The Association for Computational Linguistics Conference (ACL)*, 2000.

[88] D. Molla, F. Rinaldi, R. Schwitter, J. Dowdall, and M. Hess. Answer extraction from technical texts. *IEEE Intelligent Systems*, 2003.

[89] C. Monz. Document retrieval in the context of question answering. In *The annual European Conference on Information Retrieval (ECIR)*, 2003.

[90] C. Monz. From document retrieval to question answering. In *Ph. D. Dissertation, Universiteit Van Amsterdam*, 2003.

[91] Y. Niu and G. Hirst. Analysis of semantic classes in medical text for question answering. *Workshop on Question Answering In Restricted Domain at ACL*, 2004.

[92] Y. Niu, G. Hirst, and G. McArthur. Answering clinical questions with role identification. *Workshop on Natural Language Processing in Biomedicine at ACL*, 2003.

[93] E. Nyberg, T. Mitamura, J. Callan, J. Carbonell, R. Frederking, K. Collins-Thompson, L. Hiyakumoto, Y. Huang, C. Huttenhower, S. Judy, J. Ko, A. Kupsc, L. V. Lita, V. Pedro, D. Svoboda, and B. V. Durme. The javelin question-answering system at trec 2003: A multi strategy approach with dynamic planning. In *Text REtrieval Conference (TREC)*, 2003.

[94] E. Nyberg, T. Mitamura, J. Carbonell, J. Callan, K. Collins-Thompson, K. Czuba, M. Duggan, L. Hiyakumoto, N. Hu, Y. Huang, J. Ko, L. V. Lita, S. Murtagh, V. Pedro, and D. Svoboda. The javelin question-answering system at trec 2002. In *Text REtrieval Conference (TREC)*, 2002.

[95] E. Nyberg, T. Mitamura, R. Frederking, V. Pedro, M. Bilotti, A. Schlaikjer, and K. Hannan. Extending the javelin qa system with domain semantics. In *American Association for Artificial Intelligence Conference (AAAI)*, 2005.

[96] K. Papineni, S. Roukos, T. Ward, and W. Zhu. Bleu: a method for automatic evaluation of machine translation. In *IBM Research Report RC22176 (W0109-022)*, 2001.

[97] M. Pasca and S. Harabagiu. The informative role of wordnet in open-domain question answering. In *NAACL Workshop on WordNet and Other Lexical Resources*, 2001.

[98] J. M. Prager, E. Brown, A. Coden, and D. Radev. Question answering by predictive annotation. In *ACM Special Interest Group on Information Retrieval Conference (SIGIR)*, 2000.

[99] J. M. Prager, J. Chu-Carroll, and K. Czuba. Use of wordnet hypernyms for answering what-is questions. In *Text REtrieval Conference (TREC)*, 2001.

[100] J. M. Prager, J. Chu-Carroll, and K. Czuba. Question answering using constraint satisfaction: Qa-by-dossier-with-contraints. In *The Association for Computational Linguistics Conference (ACL)*, 2004.

[101] J. M. Prager, D. Radev, E. Brown, A. Coden, and V. Samn. The use of predictive annotation for question answering in trec8. In *Text REtrieval Conference (TREC)*, 1999.

[102] J. M. Prager, D. Radev, and K. Czuba. Answering what-is questions by virtual annotation. In *Human Language Technology Conference(HLT)*, 2001.

[103] H. Raghavan and J. Allan. Using part-of-speech patterns to reduce query ambiguity. In *ACM Special Interest Group on Information Retrieval Conference (SIGIR)*, 2002.

[104] H. Raghavan, J. Allan, and A. McCallum. An exploration of entity models, collective classification and relation description. In *Workshop on Link Analysis and Group Detection, LinkKDD2004 in conjunction with ACM SIGKDD*, 2004.

[105] D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *The Association for Computational Linguistics Conference (ACL)*, 2002.

[106] D. Ravichandran, A. Ittycheriah, and S. Roukos. Automatic derivation of surface text patterns for a maximum entropy based question answering system. In *Human Language Technology Conference - North American chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2003.

[107] F. Rinaldi, J. Dowdall, G. Schneider, and A. Persidis. Answering questions in the genomics domain. *Workshop on Question Answering In Restricted Domain at ACL*, 2004.

[108] I. Roberts and R. Gaizauska. Evaluating passage retrieval approaches for question answering. In *The annual European Conference on Information Retrieval (ECIR)*, 2004.

[109] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *International Conference on Machine Learning (ICML)*, 2001.

[110] R. Soricut and E. Brill. Automatic question answering: Beyond the factoid. In *Human Language Technology Conference - North American chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2004.

[111] M. Soubbotin and S. Soubbotin. Patterns of potential answer expressions as clues to the right answer. In *Text REtrieval Conference (TREC)*, 2001.

[112] R. Srihari and W. Li. Information extraction supported question answering. In *Text REtrieval Conference (TREC)*, 1999.

[113] R. Srihari, C. Neiu, and W. Li. A hybrid approach for named entity and sub-type tagging. In *ANLP-NAACL*, 2000.

[114] V. Stoyanov, C. Cardie, and J. Wiebe. Multi-perspective question answering using the opqa corpus. In *Human Language Technology Conference - Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, 2005.

[115] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *ACM Special Interest Group on Information Retrieval Conference (SIGIR)*, 2003.

[116] E. Terra and C. L. A. Clarke. Comparing query formulation and lexical affinity replacements in passage retrieval. In *In ELECTRA: Methodologies and Evaluation of Lexical Cohesion Techniques in Real-World Applications, SIGIR Workshop*, 2005.

[117] M. Thelen and E. Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *EMNLP*, 2002.

[118] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistic. *Technical Report, Department of Bio-statistics, Stanford University*, 2000.

[119] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistic. *J. Royal. Statist. Soc. B.*, 2001.

[120] C. van Rijsbergen. Information retrieval. In *Butterworths*, 1979.

[121] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 1963.

[122] E. M. Voorhees. The trec-8 question answering track report. In *Text REtrieval Conference (TREC)*, 1999.

[123] E. M. Voorhees. Overview of the trec-9 question answering track. In *Text REtrieval Conference (TREC)*, 2001.

[124] E. M. Voorhees. Overview of the trec 2002 question answering track. In *Text REtrieval Conference (TREC)*, 2002.

[125] E. M. Voorhees. Overview of the trec 2003 question answering track. In *Text REtrieval Conference (TREC)*, 2003.

[126] E. M. Voorhees. Overview of the trec 2004 question answering track. In *Text REtrieval Conference (TREC)*, 2004.

[127] E. M. Voorhees. Overview of the trec 2005 question answering track. In *Text REtrieval Conference (TREC)*, 2005.

[128] J.-R. Wen and H.-J. Zhang. *Information Retrieval and Clustering*, volume 11 of *Network Theory and Applications*, chapter Query Clustering in the Web Context. Klower Academic Publishers, 2003.

[129] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. 2005.

[130] W. A. Woods, S. J. Green, P. Martin, and A. Houston. Aggressive morphology and lexical relations for query expansion. In *Text REtrieval Conference (TREC)*, 2001.

[131] J. Xu, A. Licuanan, J. May, S. Miller, and R. Weischedel. Trec 2002 qa at bbn: Answer selection and confidence estimation. In *Text REtrieval Conference (TREC)*, 2002.

[132] J. Xu, A. Licuanan, and R. Weischedel. Trec 2003 qa at bbn: Answering definitional questions. In *Text REtrieval Conference (TREC)*, 2003.

[133] J. Xu, R. M. Weischedel, and A. Licuanan. Evaluation of an extraction-based approach to answering definitional questions. In *ACM Special Interest Group on Information Retrieval Conference (SIGIR)*, 2004.

[134] Y. Yang and J. Pederson. Feature selection in statistical learning of text categorizatio n. In *International Conference on Machine Learning (ICML)*, 1997.

[135] H. Yu and D. Kaufman. A cognitive evaluation of four online search engines for answering definitional questions posed by physicians. In *Pacific Symposium on Biocomputing*, 2007.

[136] D. Zhang and W. S. Lee. Question classification using support vector machines. In *ACM Special Interest Group on Information Retrieval Conference (SIGIR)*, 2003.

[137] P. Zweigenbaum. Question answering in biomedicine. *Workshop on Natural Language Processing for Question Answering at EACL*, 2003.