Instance-Based Question Answering: A Data-Driven Approach

Lucian Vlad Lita
Carnegie Mellon University
llita@cs.cmu.edu

Jaime Carbonell
Carnegie Mellon University
jgc@cs.cmu.edu

Abstract

Anticipating the availability of large question-answer datasets, we propose a principled, data-driven Instance-Based approach to Question Answering. Most question answering systems incorporate three major steps: classify questions according to answer types, formulate queries for document retrieval, and extract actual answers. Under our approach, strategies for answering new questions are directly learned from training data. We learn models of answer type, query content, and answer extraction from clusters of similar questions. We view the answer type as a distribution, rather than a class in an ontology. In addition to query expansion, we learn general content features from training data and use them to enhance the queries. Finally, we treat answer extraction as a binary classification problem in which text snippets are labeled as correct or incorrect answers. We present a basic implementation of these concepts that achieves a good performance on TREC test data.

1 Introduction

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 (Voorhees, 2003). Starting from successful pipeline architectures (Moldovan et al., 2000; Hovy et al., 2000), QA systems have responded to changes in the nature of the QA task by incorporating knowledge resources (Hermjakob et al., 2000; Hovy et al., 2002), handling additional types of questions, employing complex reasoning mechanisms (Moldovan et al., 2003; Nyberg et al., 2003), tapping into external data sources such as the Web, encyclopedias, databases (Dumais et al., 2002; Xu et al., 2003), and merging multiple agents and strategies into meta-systems (Chu-Carroll et al., 2003; Burger et al., 2002).

In recent years, learning components have started to permeate Question Answering (Clarke et al., 2003; Ravichandran et al., 2003; Echihabi and Marcu, 2003). 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 (Girju et al., 2003; Fleischman et al., 2003). Moreover, Question Answering is expanding to different languages (Magnini et al., 2003) and domains other than news stories (Zweigenbaum, 2003). These trends suggest the need for principled, statistically based, easily re-trainable, language independent QA systems that take full advantage of large amounts of training data.

We propose an *instance-based*, *data-driven* approach to Question Answering. Instead of classifying questions according to limited, predefined ontologies, we allow training data to shape the strategies for answering new questions. Answer models, query content models, and extraction models are also learned directly from training data. We present a basic implementation of these concepts and evaluate the performance.

2 Motivation

Most existing Question Answering systems classify new questions according to static ontologies. These ontologies incorporate human knowledge about the expected answer (e.g. date, location, person), answer type granularity (e.g. date, year, century), and very often semantic information about the question type (e.g. birth date, discovery date, death date).

While effective to some degree, these ontologies are still very small, and inconsistent. Considerable manual effort is invested into building and maintaining accurate ontologies even though answer types are arguably not always disjoint and hierarchical in nature (e.g. "*Where is the corpus callosum?*" expects an answer that is both location and body part).

The most significant drawback is that ontologies are not standard among systems, making individual

component evaluation very difficult and re-training for new domains time-consuming.

2.1 Answer Modeling

The task of determining the answer type of a question is usually considered a *hard* [1] decision problem: questions are classified according to an answer ontology. The classification (location, person's name, etc) is usually made 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 (Harabagiu et al., 2000) or full-fledged planning (Nyberg et al., 2003) to allow for potential answer type re-classification.

However, most questions can have multiple answer types as well as specific answer type *distributions*. The following questions 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 fi fties

However, it can be argued that *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* most likely for the third question. In fact, although the three questions can be answered by various temporal expressions, the distributions over these expressions are 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.

2.2 Document Retrieval

When faced with a new question, QA systems usually generate few, carefully expanded queries which produce ranked lists of documents. The retrieval step, which is very critical in the QA process, does not take full advantage of context information. However, similar questions with known answers *do* share context information in the form of lexical and structural features present in relevant documents. For example all questions of the type "*When was* **X** *born?*" find their answers in documents which often contain words such as "*native*" or "*record*", phrases such as "*gave birth to* **X**", and sometimes even specific parse trees.

Most IR research in Question Answering is focused on improving query expansion and structuring queries in order to take advantage of specific document pre-processing. In addition to automatic query expansion for QA (Yang et al., 2003), queries are optimized to take advantage of expansion resources and document sources. Very often, these optimizations are performed offline, based on the type of question being asked.

Several QA systems associate this type of information with question ontologies: upon observing questions of a certain type, specific lexical features are sought in the retrieved documents. These features are not always automatically learned in order to be used in query generation. Moreover, systems are highly dependent on specific ontologies and become harder to re-train.

2.3 Answer Extraction

Given a set of relevant documents, the answer extraction step consists of identifying snippets of text or exact phrases that answer the question. Manual approaches to answer extraction have been moderately successful in the news domain. Regular expressions, rule and pattern-based extraction are among the most efficient techniques for information extraction. However, because of the difficulty in extending them to additional types of questions, learning methods are becoming more prevalent.

Current systems (Ravichandran et al., 2003) already employ traditional information extraction and machine learning for extracting answers from relevant documents. Boundary detection techniques, finite state transducers, and text passage classification are a few methods that are usually applied to this task.

The drawback shared by most statistical answer extractors is their reliance on predefined ontologies. They are often tailored to expected answer types and require type-specific resources. Gazetteers, encyclopedias, and other resources are used to generate type specific features.

3 Related Work

Current efforts in data acquisition for Question Answering are becoming more and more common. (Girju et al., 2003) propose a supervised algorithm for part-whole relations extraction. (Fleischman et al., 2003) also propose a supervised algorithm that uses part of speech patterns and a large corpus to extract semantic relations for *Who-is* type questions. 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 (Chu-Carroll et al., 2003) uses

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

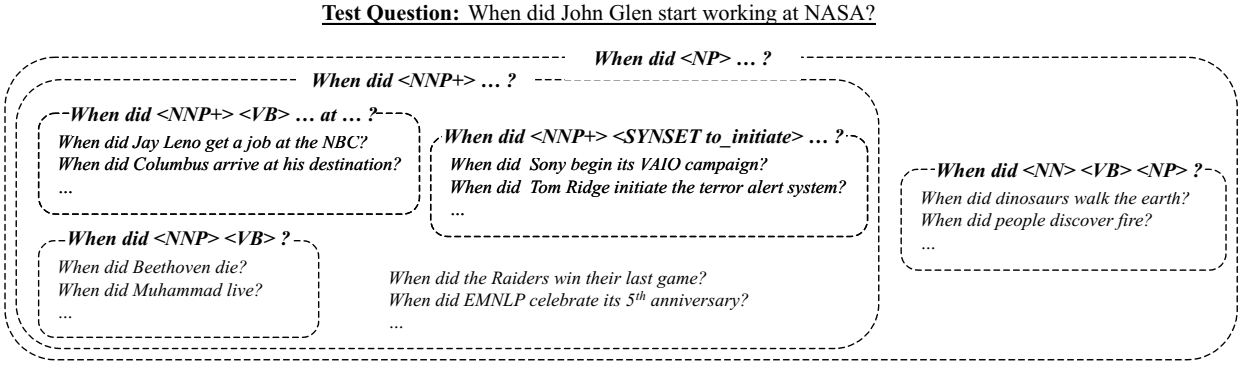

Figure 1: Neighboring questions are clustered according to features they share.

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 (Ravichandran et al., 2003) 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 (Dumais et al., 2002): high performance retrieval engines and large corpora contribute to the fact that the most redundant entity is very often the correct answer.

Predictive annotation (Prager et al., 1999) 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, (Agichtein et al., 2001) propose a method for learning query transformations in order to improve answer retrieval. The method involves learning phrase features for question classification. (Wen and Zhang, 2003) address the problem of query clustering based on semantic similarity and analyze several applications such as query re-formulation and index-term selection.

4 An Instance-Based Approach

This paper presents a data driven, instance-based approach for Question Answering. We adopt the view that strategies required in answering new questions can be directly learned from similar training examples (question-answer pairs). Consider a multi-dimensional space, determined by features extracted from training data. Each training question is represented as a data point in this space. Features can range from lexical n-grams to parse trees elements, depending on available processing.

Each test question is also projected onto the feature space. Its neighborhood consists of training instances that share a number of features with the new data point. Intuitively, each neighbor is similar in some fashion to the new question. The obvious next step would be to learn from the entire neighborhood - similar to KNN classification. However, due to the sparsity of the data and because different groups of neighbors capture different aspects of the test question, we choose to cluster the neighborhood instead. Inside the neighborhood, we build individual clusters based on internal similarity. Figure 1 shows an example of neighborhood clustering. Notice that clusters may also have different granularity - i.e. can share more or less features with the new question.

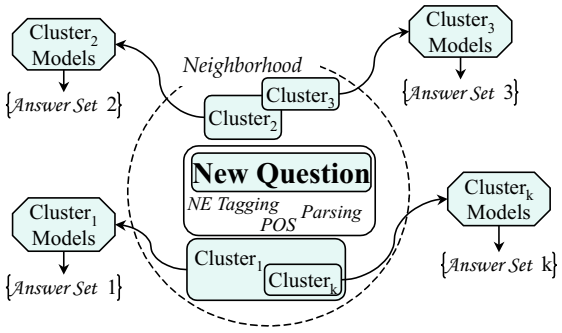

Figure 2: The new question is projected onto the multi-dimensional feature space. A set of neighborhood clusters are identified and a model is dynamically built for each of them. Each model is applied to the test question in order to produce its own set of candidate answers.

By clustering the neighborhood, we set the stage for supervised methods, provided the clusters are sufficiently dense. The goal is to learn models that explain individual clusters. A model explains the data if it successfully answers questions from its corresponding cluster. For each cluster, a models is constructed and tailored to the local data. Models generating high confidence answers are applied to the new question to produce answer candidates (Figure 2) Since the test question belongs to

multiple clusters, it benefits from different answer-seeking strategies and different granularities.

Answering clusters of similar questions involves several steps: learning the distribution of the expected answer type, learning the structure and content of queries, and learning how to extract the answer. Although present in most systems, these steps are often static, manually defined, or based on limited resources (section 2). This paper proposes a set of trainable, cluster-specific models:

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.

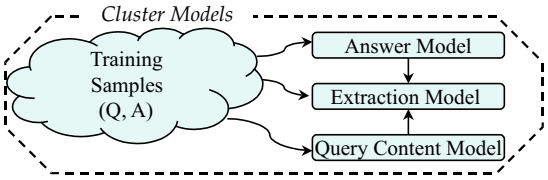

Figure 3: Three cluster-specific components are learned in order to better retrieve relevant documents, model the expected answer, and then extract it from raw text. Local question-answer pairs (Q,A) are used as training data.

These models are derived directly from cluster data and collectively define a focused strategy for finding answers to similar questions (Figure 3).

4.1 The Answer Model

Learning cluster-specific answer type distributions is useful not only in terms of identifying answers in running text but also in answer ranking. A probabilistic approach has the advantage of postponing answer type decisions from early in the QA process until answer extraction or answer ranking. It also has the advantage of allowing training data to shape the expected structure of answers.

The answer modeling task consists of learning specific answer type distributions for each cluster of questions. Provided enough data, simple techniques such as constructing finite state machines or learning regular expressions are sufficient. The principle can also be applied to current answer ontologies by replacing the hard classification with a distribution over answer types.

For high-density clusters, the problem of learning the expected answer type is reduced to learning possible answer types and performing a reliable frequency count. However, very often clusters are sparse (e.g. are based on rare features) and a more reliable method is required. k-nearest training data points $Q_1..Q_k$ can be used in order to estimate the probability that the test question q will observe an answer type α_j:

$$P(\alpha_j, q) = \mu \cdot \sum_{i=0}^{k} P(\alpha_j|Q_i) \cdot \delta(q, Q_i) \quad (1)$$

where $P(\alpha_j, Q_i)$ is the probability of observing an answer of type α_j when asking question Q_i. $\delta(q, Q_i)$ represents a distance function between q and Q_i, and μ is a normalizing factor over the set of all viable answer types in the neighborhood of q.

4.2 The Query Content Model

Current Question Answering systems use IR in a straight-forward fashion. Query terms are extracted and then expanded using statistical and semantic similarity measures. Documents are retrieved and the top K are further processed. This approach describes the traditional IR task and does not take advantage of specific constraints, requirements, and rich context available in the QA process.

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 (Hermjakob et al., 2002) which yield better queries when added to simple keyword-based queries. The **Query Content Model** is a *cluster-specific* collection of content features that generate the best document set (Table 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 1: 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.

For training, simple keyword-based queries are run through a retrieval engine in order to produce a set of potentially relevant documents. Features

(n-grams and paraphrases) are extracted and scored based on their co-occurrence with the correct answer. More specifically, consider a positive class: documents which contain the correct answer, and a negative class: documents which do not contain the answer. We compute the average mutual information $I(C; F_i)$ between a class of a document, and the absence or presence of a feature f_i in the document (McCallum and Nigam, 1998). We let C be the class variable and F_i the feature variable:

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

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 candidate queries by individually adding selected features (e.g. table 1) to the expanded word-based query. The resulting candidate queries are subsequently run through a retrieval engine and scored based on the number of passages containing correct answers (precision). The content features found in the top u candidate queries are included in the Query Content Model.

The Content Model is *cluster specific* and not *instance specific*. It does not replace traditional query expansion - both methods can be applied simultaneously to the test questions: specific keywords are the basis for traditional query expansion and clusters of similar questions are the basis for learning additional content conducive to better document retrieval. Through the Query Content Model we allow shared context to play a more significant role in query generation.

4.3 The Extraction Model

During training, documents are retrieved for each question cluster and a set of one-sentence passages containing a minimum number of query terms is selected. The passages are then transformed into feature vectors to be used for classification. The features consist of n-grams, paraphrases, distances between keywords and potential answers, simple statistics such as document and sentence length, part of speech features such as required verbs etc. More extensive sets of features can be found in information extraction literature (Bikel et al., 1999).

Under our data-driven approach, answer extraction consists of deciding the correctness of candidate passages. The task is to build a model that accepts snippets of text and decides whether they contain a correct answer.

A classifier is trained for each question cluster. When new question instances arrive, the already trained cluster-specific models are applied to new, relevant text snippets in order to test for correctness. We will refer to the resulting classifier scores as answer confidence scores.

5 Experiments

We present a basic implementation of the instance-based approach. The resulting QA system is fully automatically trained, without human intervention.

Instance-based approaches are known to require large, dense training datasets which are currently under development. Although still sparse, the subset of all *temporal* questions from the TREC 9-12 (Voorhees, 2003) datasets is relatively dense compared to the rest of the question space. This makes it a good candidate for evaluating our instance-based QA approach until larger and denser datasets become available. It is also broad enough to include different question structures and varying degrees of difficulty and complexity such as:

- *"When did Beethoven die?"*
- *"How long is a quarter in an NBA game?"*
- *"What year did General Montgomery lead the Allies to a victory over the Axis troops in North Africa?"*

The 296 temporal questions and their corresponding answer patterns provided by NIST were used in our experiments. The questions were processed with a part of speech tagger (Brill, 1994) and a parser (Collins, 1999).

The questions were clustered using template-style frames that incorporate lexical items, parser labels, and surface form flags (Figure 1). Consider the following question and several of its corresponding frames:

"When did Beethoven die?"

when did <*NNP*> die
when did <*NNP*> <*VB*>
when did <*NNP*> <*Q*>
when did <*NP*> <*Q*>
when did <*Q*>

where <*NNP*>,<*NP*>,<*VB*>,<*Q*> denote: proper noun, noun phrase, verb, and generic question term sequence, respectively. Initially, frames are generated exhaustively for each question. Each frame that applies to more than three questions is then selected to represent a specific cluster.

One hundred documents were retrieved for each query through the Google API

(*www.google.com/api*). Documents containing the full question, question number, references to *TREC*, *NIST*, *AQUAINT*, *Question Answering* and other similar problematic content were filtered out.

When building the Query Content Model keyword-based queries were initially formulated and expanded. From the retrieved documents a set of content features (n-grams and paraphrases) were selected through average mutual information. The features were added to the simple queries and a new set of documents was retrieved. The enhanced queries were scored and the corresponding top 10 n-grams/paraphrases were included in the Query Content Model. The maximum n-gram and paraphrase size for these features was set to 6 words.

The Extraction Model uses a support vector machine (SVM) classifier (Joachims, 2002) with a linear kernel. The task of the classifier is to decide if text snippets contain a correct answer. The SVM was trained on features extracted from one-sentence passages containing at least one keyword from the original question. The features consist of: distance between keywords and potential answers, keyword density in a passage, simple statistics such as document and sentence length, query type, lexical n-grams (up to 6-grams), and paraphrases.

We performed experiments using leave-one-out cross validation. The system was trained and tested without any question filtering or manual input. Each cluster produced an answer set with corresponding scores. Top 5 answers for each instance were considered by a mean reciprocal rank (MRR) metric over all N questions: $MRR_N = \sum_{i=0}^{N} \frac{1}{rank_i}$, where $rank_i$ refers to the first correct occurrence in the top 5 answers for question i. While not the focus of this paper, answer clustering algorithms are likely to further improve performance.

6 Results

The most important step in our instance-based approach is identifying clusters of questions. Figure 4 shows the question distribution in terms of number of clusters. For example: 30 questions belong to exactly 3 clusters. The number of clusters corresponding to a question can be seen as a measure of how common the question is - the more clusters a question has, the more likely it is to have a dense neighborhood.

The resulting MRR is 0.447 and 61.5% questions have correct answers among the first five proposed answers. This translates into results consistently above the sixth highest score at each TREC 9-12. Our results were compared directly to the top performing systems' results on the same temporal question test set.

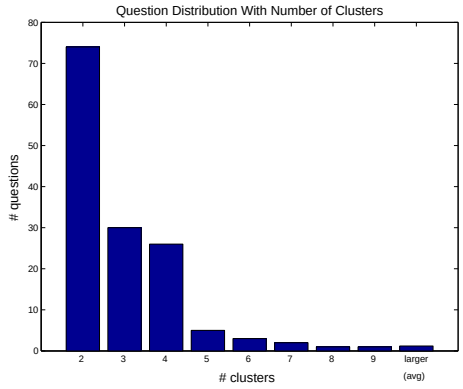

Figure 4: Question distribution - each bar shows the number of questions that belong to exactly c clusters.

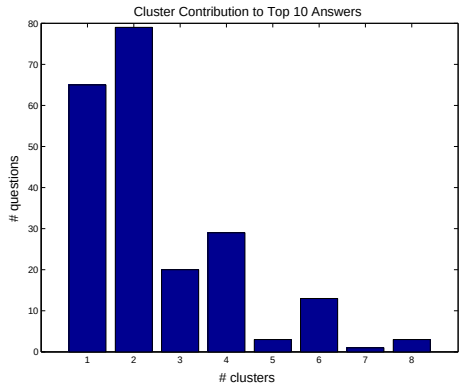

Figure 5: Number of clusters that contribute with correct answers to the final answer set - only the top 10 answers were considered for each question.

Figure 5 shows the degree to which clusters produce correct answers to test questions. Very often, more than one cluster contributes to the final answer set, which suggests that there is a benefit in clustering the neighborhood according to different similarity features and granularity.

It is not surprising that cluster size is not correlated with performance (Figure 6). The overall strategy learned from the cluster "*When did <NP> die?*" corresponds to an MRR of 0.79, while the strategy learned from cluster "*How <Q>?*" corresponds to an MRR of 0.13. Even if the two clusters generate strategies with radically different performance, they have the same size - 10 questions are covered by each cluster.

Figure 7 shows that performance is correlated with answer confidence scores. The higher the confidence threshold the higher the precision (MRR) of the predicted answers. When small, unstable clusters are ignored, the predicted MRR improves considerably. Small clusters tend to produce unsta-

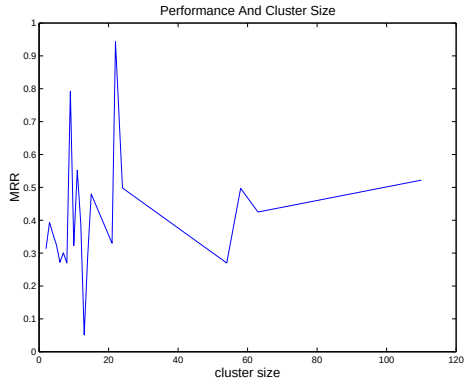

Figure 6: Since training data is not uniformly distributed in the feature space, cluster size is not well correlated with performance. A specifi c cardinality may represent a small and dense part cluster, or a large and sparse cluster.

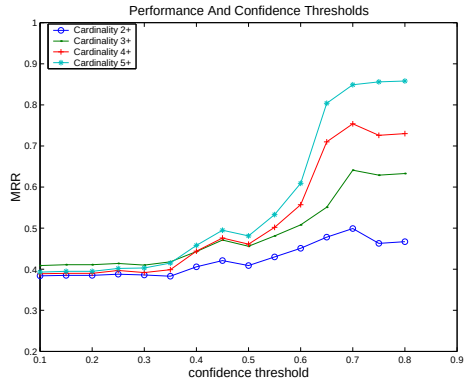

Figure 7: MRR of predicted answers varies with answer confi dence thresholds. There is a tradeoff between confi - dence threshold and MRR . The curves represent different thresholds for minimum cluster size.

ble strategies and have extremely low performance. Often times structurally different but semantically equivalent clusters have a higher cardinality and much better performance. For example, the cluster "*What year did <NP> die?*" has cardinality 2 and a corresponding MRR of $zero$. However, as seen previously, the cluster "*When did <NP> die?*" has cardinality 10 and a corresponding MRR of 0.79.

Table 2 presents an intuitive cluster and the top n-grams and paraphrases with most information content. Each feature has also a corresponding average mutual information score. These particular content features are intuitive and highly indicative of a correct answer. However, in sparse clusters, the content features have less information content and are more vague. For example, the very sparse cluster "*When was <Q>?*" yields content features such as "*April*", "*May*", "*in the spring of*", "*back in*" which only suggest broad temporal expressions.

Cluster: When did <QTERM> die?			
N-grams		*Paraphrases*	
0.81	his death in	0.80	<Q> died in
0.78	died on	0.78	<Q> died
0.77	died in	0.68	<Q> died on
0.75	death in	0.58	<Q> died at
0.73	of death	0.38	<Q> , who died
0.69	to his death	0.38	<Q> dies
0.66	died	0.38	<Q> died at the age of
0.63	, born on	0.38	<Q> , born
0.63	date of death	0.35	<Q> 's death on

Table 2: Query Content Model: learning n-grams and paraphrases for class "*When did <NP> die?*", where *<Q>* refers to a phrase in the original question.

7 Conclusions

This paper presents an principled, statistically based, instance-based approach to Question Answering. Strategies and models required for answering new questions are directly learned from training data. Since training requires very little human effort, relevant context, high information query content, and extraction are constantly improved with the addition of more question-answer pairs.

Training data is a critical resource for this approach - clusters with very few data points are not likely to generate accurate models. However, research efforts involving data acquisition are promising to deliver larger datasets in the near future and solve this problem. We present an implementation of the instance-based QA approach and we evaluate it on temporal questions. The dataset is of reasonable size and complexity, and is sufficiently dense for applying instance-based methods. We performed leave-one-out cross validation experiments and obtained an overall mean reciprocal rank of 0.447. 61.5% of questions obtained correct answers among the top five which is equivalent to a score in the top six TREC systems on the same test set.

The experiments show that strategies derived from very small clusters are noisy and unstable. When larger clusters are involved, answer confidence becomes correlated with higher predictive performance. Moreover, when ignoring sparse data, answering strategies tend to be more stable. This supports the need for more training data as means to improve the overall performance of the data driven, instance based approach to question answering.

8 Current & Future Work

Data is the single most important resource for instance-based approaches. Currently we are exploring large-scale data acquisition methods that can provide the necessary training data density for

most question types, as well as the use of trivia questions in the training process.

Our data-driven approach to Question Answering has the advantage of incorporating learning components. It is very easy to train and makes use of very few resources. This property suggests that little effort is required to re-train the system for different domains as well as other languages. We plan to apply instance-based QA to European languages and test this hypothesis using training data acquired through unsupervised means.

More effort is required in order to better integrate the cluster-specific models. Strategy overlap analysis and refinement of local optimization criteria has the potential to improve overall performance under time constraints.

References

E. Agichtein, S. Lawrence, and L. Gravano. 2001. Learning search engine specific query transformations for question answering. *WWW*.

D. Bikel, R. Schwartz, and R. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*.

E. Brill. 1994. Some advances in rule-based part of speech tagging. *AAAI*.

J. Burger, L. Ferro, W. Greiff, J. Henderson, M. Light, and S. Mardis. 2002. Mitre's qanda at trec-11. *TREC*.

J. Chu-Carroll, K. Czuba, J. Prager, and A. Ittycheriah. 2003. In question answering, two heads are better than one. *HLT-NAACL*.

C. Clarke, G. Cormack, G. Kemkes, M. Laszlo, T. Lynam, E. Terra, and P. Tilker. 2003. Statistical selection of exact answers. *TREC*.

M. Collins. 1999. Head-driven statistical models for natural language parsing. *Ph.D. Disertation*.

S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. 2002. Web question answering: Is more always better? *SIGIR*.

A. Echihabi and D. Marcu. 2003. A noisy channel approach to question answering. *ACL*.

M. Fleischman, E. Hovy, and A. Echihabi. 2003. Offline strategies for online question answering: Answering questions before they are asked. *ACL*.

R. Girju, D. Moldovan, and A. Badulescu. 2003. Learning semantic constraints for the automatic discovery of part-whole relations. *HLT-NAACL*.

S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, R. Bunescu, R. Girju, V. Rus, and P. Morarescu. 2000. Falcon: Boosting knowledge for answer engines. *TREC*.

U. Hermjakob, E. Hovy, and C. Lin. 2000. Knowledge-based question answering. *TREC*.

Ulf Hermjakob, Abdessamad Echihabi, and Daniel Marcu. 2002. Natural language based reformulation resource and web exploitation for question answering. *TREC*.

E. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C.Y. Lin. 2000. Question answering in webclopedia. *TREC*.

E. Hovy, U. Hermjakob, C. Lin, and D. Ravichandran. 2002. Using knowledge to facilitate factoid answer pinpointing. *COLING*.

T. Joachims. 2002. Learning to classify text using support vector machines. *Disertation*.

B. Magnini, S. Romagnoli, A. Vallin, J. Herrera, A. Penas, V. Peiado, F. Verdejo, and M. de Rijke. 2003. The multiple language question answering track at clef 2003. *CLEF*.

A. McCallum and K. Nigam. 1998. A comparison of event models for naive bayes text classification. *AAAI, Workshop on Learning for Text Categorization*.

D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Girju, R. Goodrum, and V. Rus. 2000. The structure and performance of an open-domain question answering system. *ACL*.

D. Moldovan, D. Clark, S. Harabagiu, and S. Maiorano. 2003. Cogex: A logic prover for question answering. *ACL*.

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. Vand Durme. 2003. A multi strategy approach with dynamic planning. *TREC*.

J. Prager, D. Radev, E. Brown, A. Coden, and V. Samn. 1999. The use of predictive annotation for question answering in trec8. *TREC*.

D. Ravichandran, A. Ittycheriah, and S. Roukos. 2003. Automatic derivation of surface text patterns for a maximum entropy based question answering system. *HLT-NAACL*.

E.M. Voorhees. 2003. Overview of the trec 2003 question answering track. *TREC*.

J.R. Wen and H.J. Zhang. 2003. Query clustering in the web context. *IR and Clustering*.

J. Xu, A. Licuanan, and R. Weischedel. 2003. Trec 2003 qa at bbn: Answering definitional questions. *TREC*.

H. Yang, T.S. Chua, S. Wang, and C.K. Koh. 2003. Structured use of external knowledge for event-based open domain question answering. *SIGIR*.

P. Zweigenbaum. 2003. Question answering in biomedicine. *EACL*.