Question Answering with LCC's CHAUCER-2 at TREC 2007

Andrew Hickl, Kirk Roberts, Bryan Rink, Jeremy Bensley, Tobias Jungen, Ying Shi, and John Williams
(DRAFT)
Language Computer Corporation
1701 North Collins Boulevard
Richardson, Texas 75080
andy@languagecomputer.com

Abstract

In TREC 2007, Language Computer Corporation explored how a new, semantically-rich framework for information retrieval could be used to boost the overall performance of the answer extraction and answer selection components featured in its CHAUCER-2 automatic question-answering (Q/A) system. By replacing the traditional keyword-based retrieval system used in (Hickl *et al.* 2006c) with a new indexing and retrieval engine capable of retrieving documents or passages based on the distribution of named entities or semantic dependencies, we were able to dramatically enhance CHAUCER-2's overall accuracy, while significantly reducing the number of of candidate answers that were considered by its Answer Ranking and Answer Validation modules.

1. Introduction

In TREC 2007, Language Computer Corporation explored how a new, semantically-rich framework for information retrieval could be used to boost the overall performance of the answer extraction and answer selection components featured in its CHAUCER-2 automatic question-answering (Q/A) system.

Unlike the keyword-based retrieval systems traditionally used by Q/A systems, CHAUCER-2 leverages a novel indexing and retrieval engine which makes it possible to retrieve documents or passages using queries that look for instances of (1) entity types recognized by a named entity recognition (NER) system, (2) semantic dependencies identified by PropBank- or NomBank-based semantic parsers, (3) semantic frames (or frame-denoting elements) recognized by a FrameNet parser, or even (4) the normalized versions of temporal or spatial expressions. Support for these new types of queries dramatically enhanced the performance of CHAUCER-2's Document and Passage Retrieval components while significantly reducing the number of candidate answers that had to be considered by its Answer Ranking and Answer Validation modules.

CHAUCER-2 also leverages a variant of the Bindings Engine (BE) first proposed by (Cafarella *et al.* 2005; Cafarella & Etzioni 2005) in order to retrieve of all of the text snippets matched by a pattern-based (or variabilized) query without having to retrieve documents using a keyword-based query. We found that use of this framework greatly both enhanced the efficiency and the recall of traditional pattern-based approaches to Q/A and allowed for the development of new libraries of precise patterns for specific question types asked in previous TREC QA evaluations.

Finally, CHAUCER-2 incorporates a new, multi-tiered Answer Type Detection (ATD) module which reduces the number of expected answer types (EATs) considered by the system for factoid or list questions, while maintaining the same high levels of precision exhibited by previous systems. Since LCC's previous ATD systems often identified a large number of spurious answer types along with the most correct answer type for a question, we developed a new back-off mechanism which forces the Q/A system to consider the most specific EATs identified for a question first; other, more general EATs were then included as search terms only when insufficient evidence was retrieved using the more specific EAT.

Taken together, we believe these three enhancements to CHAUCER-2's core retrieval capabilities allowed us to develop a battery of high-precision, low-recall Answer Retrieval strategies which could be run independently of the traditional entity-based Q/A strategies currently being used by LCC's FERRET (Hickl *et al.* 2006a) and CHAUCER-1 (Hickl *et al.* 2006c) question-answering systems. In order to maximize the value of these individual strategies, we re-cast the new CHAUCER-2 Q/A pipeline developed for the TREC 2007 evaluations as a cascade of end-to-end Q/A systems which were tasked with answering questions in order of their expected precision for a particular question type.

The rest of this paper is organized in the following way. Section 2 presents a brief overview of the architecture of the CHAUCER-2 system. Section 3 presents details of CHAUCER-2's core factoid Q/As system, while Section 4 describes the system for answering list questions, and Section 5 describes the techniques used to answer "other" questions. Results from this year's official evaluation are discussed in Section 6, while Section 7 summarizes our conclusions.

2. The CHAUCER-2 Question-Answering System

This section describes the architecture of the CHAUCER-2 question-answering system used to answer factoid and list

questions for the TREC 2007 QA Track Main Task. The architecture of CHAUCER-2 is presented in Figure 1.

Target Processing

Targets in CHAUCER-2 are initially submitted to a *Target Type Classification* module which uses a version of the Maximum Entropy-based classifier introduced in (Hickl *et al.* 2006c) in order to categorize series targets into one of a set of semantic categories taken from the large ontology of semantic types recognized by LCC's CICEROLITE named entity recognition system. As with our TREC 2006 work, targets were classified into one of six target types, including: (1) PEOPLE (e.g. *Warren Moon*), (2) ORGANIZATIONS (*American Enterprise Institute*), (3) LOCATIONS (*Amazon River*), (4) EVENTS (*1991 eruption of Mount Pinatubo*), (5) AUTHORED WORK (*The Daily Show*), or (6) GENERIC NOUNS (*avocados*).

Classified targets were then sent to a *Discovery of Essential Information* module which leveraged sets of event, attribute, and relationship extractors created prior to the TREC 2007 evaluations for each individual target type category using LCC's CICEROCUSTOM open-domain, customizable information extraction system. In addition to these sets of custom extractors, CHAUCER-2 also used sets of heuristics in order to extract information related to targets from a number of "authoritative sources" available on the WWW, including *imdb.com, nndb.com, iplpotus.com, s9.com,* and *wikipedia.org*. Once each of these four extraction strategies were run for an individual target, output was then cast in a structured form and stored in a database (referred to as the *Factoid Database*).

In order to ensure that the *Factoid Database* contained a minimum of contradictory and/or redundant information, all new information added to the database was first sent to a *Content Validation* module, which followed (Hickl & Harabagiu 2007) in using the output of systems for recognizing textual entailment (Hickl & Bensley 2007) and textual contradiction (Harabagiu, Hickl, & Lacatusu 2006) in order to determine when newly-discovered facts could be either inferred from – or were directly contradicted by – knowledge already stored in the database.

Once the *Factoid Database* was populated for each target, a select set database fields were then sent to a *QUAB Generation* module in order to generate sets of question-answer pairs which could be used by CHAUCER-2's other downstream *Answer Selection* and *Answer Validation* modules.

Question Processing

Following *Target Processing*, each question in a series was then sent to a series of *Question Processing* modules which annotated individual questions with the lexico-semantic information needed to generate queries for each of the individual *Answer Retrieval* strategies employed by CHAUCER-2.

As with our TREC 2006 system, questions were initially sent to a *Question Annotation* module which (1) identified token and collocation boundaries, performed (2) part-of-speech tagging and (3) named entity annotation (using LCC's CICEROLITE named entity recognition system), and (4) resolved instances of pronominal and nominal coreference (using the knowledge-lean, heuristic approach first introduced in (Hickl *et al.* 2006c)). Questions were then sent to a *Semantic Parsing* module, which identified semantic dependencies using LCC's own PropBank-, NomBank-, and FrameNet-based semantic parsers.

Factoid and list questions were then sent to a new *Answer Type Detection* module which followed (Li & Roth 2002) and (Chakrabarti, Krishnan, & Das 2005) in using a multiple Maximum Entropy-based classifiers in order to identify the expected answer type (EAT) of the question from LCC's answer type hierarchy.

Once annotation and answer type detection were complete, CHAUCER-2 sent questions to a *Query Formulation* module responsible for (1) extracting keywords and phrases, (2) identifying synonymous terms that could be used to augment a query, and (3) transforming questions into the particular kinds of queries required by each of the system's *Answer Retrieval* strategies.

Document Preprocessing and Retrieval

CHAUCER-2 employed the same document preprocessing framework introduced in (Hickl *et al.* 2006c). As with our TREC 2006 submission, we preprocessed the AQUAINT corpus with four types of information. First, we used an in-house implementation of the Collins parser to provide a full syntactic parse of every document in the AQUAINT-2 corpus; documents in the larger (and "noisier") BLOG-06 corpus were parsed using an in-house chunk parser. Second, we used three different semantic parsers in order to identify semantic dependencies imposed by both verbal and nominalized predicates. In addition to LCC's PropBank and NomBank parsers, we also used LCC's FrameNet-based semantic parser to identify instances FrameNet frames in natural language texts; a separate role classifier was used to identify roles associated each FrameNet frame.[1] Third, we used LCC's CICEROLITE named entity recognition system in order to classify more than 300 different types of names found in the corpus. We also used more than 500 lexicons and gazetteers derived from web-based resources in order to tag additional name types not covered by CICEROLITE. Finally, we used LCC's PINPOINT temporal normalization system (Lehmann *et al.* 2005) in order to map temporal expressions found in documents to a standardized (ISO 8601) format.

Following annotation, we indexed the AQUAINT-2 and BLOG-06 corpora using a customized version of the Lucene information retrieval engine. In addition to retrieving documents (and passages) based on literal strings and stemmed words, CHAUCER-2 was able also retrieve documents based on a wide range of semantic annotations made available by LCC's annotation components, including (1) entity types from LCC's CICEROLITE, (2) predicate-argument relationships from LCC's PropBank and NomBank parsers, (3) se-

[1] The AQUAINT-2 corpus was processed using semantic parsers which had been previously trained on data that had been fully parsed; documents in the BLOG-06 corpus were processed using parsers that had been trained on the output of a chunk parser.

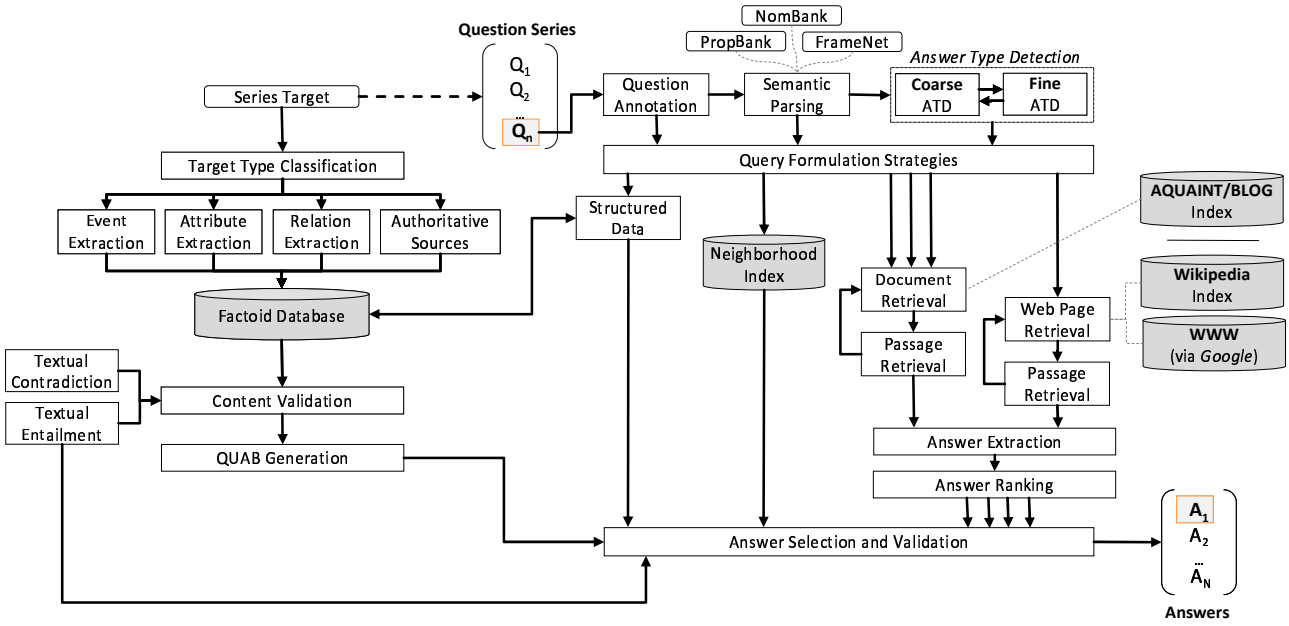

Figure 1: Architecture of the CHAUCER-2 Question-Answering System

mantic frames, frame roles, and frame-denoting elements recognized by LCC's FrameNet parser, or (4) any of the events, attributes, or relations extracted by LCC's CICERO-CUSTOM.

Answer Retrieval and Extraction

CHAUCER-2 leverages a cascade of three separate factoid *Answer Retrieval and Extraction* strategies in order to identify the best answer to each question in a series. Strategies developed for this year's TREC include: (1) a *Structured Data* strategy which identifies answers from the information stored in the *Factoid Database*, (2) a *Pattern-based* strategy which leverages a variant of the Bindings Engine (BE) first proposed by (Cafarella & Etzioni 2005; Cafarella *et al.* 2005) in order to retrieve all of the text snippets matched by a pattern-based query, and (3) a traditional *Entity-based* strategy identifies candidate answers based on the distribution of entity types associated with an expected answer type. (Details of each of these three strategies are presented in Section 3.)

Although previous versions of CHAUCER (Harabagiu *et al.* 2005; Hickl *et al.* 2006c) have sought to identify likely candidate answers by combining the output of multiple Q/A strategies, CHAUCER-2 considers answers returned by its four Q/A engines in a fixed order. If no answers are returned by the *Structured Data* strategy, then questions are sent to the *Neighborhood-based* strategy; likewise, if no answers above a fixed confidence threshold are returned by the *Neighborhood-based* strategy, CHAUCER-2 defaults to using the *Entity-based* strategy in order to find answers.

Answer Validation

As with our TREC 2006 submission, CHAUCER-2 employs a *Answer Selection and Validation* module in order to identify the best answer when multiple candidate answers are returned by one (or more) *Answer Extraction* strategies. Following *Answer Extraction*, the top five candidate answers identified by each strategy are then sent to a *Candidate Answer Re-ranking* module which uses a Maximum Entropy-based re-ranker (based on (Ravichandran, Hovy, & Och 2003) in order to provide a single ranked list of candidate answers for a particular question. The re-ranked list of answers were then sent to a final *Answer Selection* module which uses the state-of-the-art textual entailment system described in (Hickl & Bensley 2007) in order to identify the single answer passage whose meaning is most likely to be entailed by the meaning of the original question.

3. Answering Factoid Questions

In this section, we describe the CHAUCER-2 system for answering factoid questions.

Question Processing

This section describes how questions were processed in CHAUCER-2.

Keyword Expansion As with the TREC 2006 version of CHAUCER, keywords extracted from each question were processed by a *Keyword Expansion* module that was designed to identify additional synonymous keywords that could be used to augment the query CHAUCER-2 used to retrieve documents. This module used a set of heuristics in order to append synonyms and alternate keywords from a database of similar terms developed by LCC for previous TREC QA evaluations.

Question Coreference We incorporated a heuristic-based *Question Coreference* module in order to resolve referring expressions found in the question series to antecedents mentioned in previous questions or in the target description. First, we used heuristics for performing name aliasing and

nominal coreference from CICEROLITE in order to identify the full referent for each partial name mention found in the question series. Next, we constructed an *antecedent list* from all of the named entities that occurred in the question series prior to the current question. Each potential antecedent and referring expression found in the series were then annotated with name class, gender, and number information available from CICEROLITE. We then used the Hobbs Algorithm (Hobbs 1978) in order to match referring expressions to candidate antecedents. When no compatible antecedent could be identified from the antecedent list, we made no further attempt to resolve the referring expression found in the question.

Answer Type Detection CHAUCER-2 follows recent work in *Answer Type Detection* (Li & Roth 2002; Chakrabarti, Krishnan, & Das 2005; Hickl *et al.* 2006c) (ATD) in using a multi-tiered classification approach to the recognition of the Expected Answer Type (EAT) of both factoid and list questions. Under our current approach, we use a three-stage approach to identifying the expected answer type of a question. First, questions are submitted to a *coarse type* ATD classifier which uses an variant of the Maximum Entropy-based classifier first introduced in (Harabagiu *et al.* 2005) in order to associate each question with one of a set of 11 coarse types. (The complete list of coarse types that were used in TREC 2007 are listed in Table 1.) Second, questions of certain selected answer types are submitted to a second, *expanded coarse type* classifier which identifies a second coarse-grained answer type which can be used to further describe the type of answer sought by the question. (The set of expanded coarse types we considered in our TREC 2007 work are presented in Table 2.) Finally, questions of each coarse type (or sub-type) are then submitted to a third set of *fine type* classifiers which map each question to one of the set of fine answer types associated with each coarse type. In our work, we have used a hierarchy of over 275 different fine entity types derivable from the more than 300 different entity types recognized by LCC's CICEROLITE. (Table 3 presents a sample of some of the fine types that were used in CHAUCER-2.)

Coarse Type	Example(s)
HUMAN	George W. Bush, Texans, State Department
LOCATION	Tajikistan, Grand Canyon, Sears Tower
ABBREVIATION	AARP, Dr.
WORK	Hamlet, Guernica
NUMERIC	55mph, £124
TEMPORAL	1945, 8 years ago
TITLE	Physician, Israeli
CONTACT-INFO	andy@languagecomputer.com
OTHER-ENTITY	Hurricane Andrew, Budweiser
OTHER-VALUE	purple, guilty
COMPLEX	–

Table 1: Coarse Answer Types used by CHAUCER-2

In our TREC 2007 work, we have found that CHAUCER-2's approach to *Answer Type Detection* hinges on the recognition of three core elements from each question: (1) the *question stem*, (2) the *predicate answer type term*, and (3) the *nominal answer type term*.

Coarse Type	Expanded Coarse Type	Example(s)
HUMAN	INDIVIDUAL	Bill Clinton, Paul McStay
	GROUP	journalists, Floridians
	ORGANIZATION	FBI, The White Stripes
LOCATION	FACILITY	MacDill AFB, Hoover Dam
	GPE	India, Los Angeles
	PHYSICAL LOCATION	Great Plains, Blue Nile

Table 2: Breakdown of HUMAN and LOCATION Coarse Answer Types into Expanded Coarse Types

Coarse Type	Fine Types
FACILITY	CASINO, MUSEUM
GPE	CITY, COUNTRY, STATE
INDIVIDUAL	ACTOR, BASEBALL-PLAYER, MILITARY-PERSON
ORGANIZATION	COMPANY, UNIVERSITY, BASEBALL-TEAM
PHYSICAL LOCATION	ISLAND, PLANET, RIVER
WORK	ALBUM, SONG, BOOK

Table 3: Examples of CHAUCER-2's Fine Answer Types.

We define a *question stem* as the word (or phrase) which signals the broadest type of information sought by the question. With most interrogatives, the question stem is equivalent to a WH-word (e.g. *who, what how*) or a WH-phrase (e.g. *how many, what book* and can be extracted heuristically from the text of a question.[2] We consider a question's *predicate answer type term* (or predicate ATT) to be any verbal predicate (or predicate nominal) which exhibits a semantic dependency with a question stem. For example, in a question like "*What civilization built the pyramids that towered over the Nile River?*", the words *built* and *towered* are both predicates, but only the predicate *built* has selects the question stem *What nation* as an argument. In contrast, we define the *nominal answer type term* (or nominal ATT) as the noun phrase (NP) in a question that can lead to the inference of a question's expected answer type (EAT). For example, in the questions *What country is Ahmadinejad president of?* and *What is Jon Bon Jovi's profession?*, we assume that words such as *country* and *profession* can be used to infer an the most appropriate answer type for these questions.

In CHAUCER-2, recognition of the *question stem*, predicate ATT, and nominal ATT were performed using a heuristic based method that was tuned on a collection of more than 6000 factoid questions which had been annotated with these three core elements.

Evaluation results for nominal ATT detection are listed in Table 4. CHAUCER-2 is least accurate on question stems that need no nominal ATT, such as *who*, *when*, and *where*. However, since these questions already derive much meaning from their stems, the downstream performance is not significantly damaged. On *what* questions, however, miss-

[2] We assume that question stem of an imperative questions like *Name books that Pamuk has written.* corresponds to the initial predicate which signals both that the statement is a request for information and the type of information that the speaker presumably seeks.

ing the nominal ATT will almost always cause the final answers to be incorrect. We found that the most common cause for missing the nominal ATT occurs in syntactic parsing or while interpreting the syntactic parse tree. For example, syntactic parsers will often mis-parse question, "*What state-of-the-art technique is being used for the newest TMNT movie?*" without the use of high performance chunking or collocation detection. In this question, CHAUCER-2 incorrectly annotates *state* as the nominal ATT instead of *technique*.

Question Stem	Total Questions	Accuracy
who/whom/whose	58	89.7
what/which	278	97.8
when	13	92.3
where	13	92.3
how	65	100
list	8	100
name	10	100
Total	**445**	**96.9**

Table 4: Nominal Answer Type Term Detection Results, by question stem, on TREC 2007 Questions.

The overall answer type detection accuracy scores for CHAUCER-2 are listed in Table 5. The final score is primarily due to the combined error of the Coarse, Human, and Location classifiers.

Type	Total Questions	Accuracy
Coarse	445	90.6%
Human	154	90.3%
Location	59	88.1%
Fine	**445**	**79.3%**

Table 5: Answer Type Detection Results on TREC 2007 Questions.

Document Retrieval CHAUCER-2 takes advantage of the same two-tiered approach to document retrieval first introduced in (Hickl *et al.* 2006c). Under this approach, output from a conservative entity-based answer extraction strategy was used in order to re-rank the top 200 documents retrieved from CHAUCER-2's standard retrieval engine.

Our TREC 2007 approach follows the same four-step approach that was implemented for our TREC 2006 system. First, we used a standard (expanded) keyword query to retrieve a total of 200 documents from the AQUAINT-2 and BLOG-06 corpora. The top 50 passages were then identified using a passage retrieval engine and submitted to CHAUCER-2's traditional entity-based answer extraction system. Passages were then re-ranked based on both (1) the density keywords extracted from the question found in each passage and (2) the distribution of entity types corresponding to the expected answer type of the question. The original set of 200 retrieved documents were then re-ranked based on the distribution of the top-ranked passages. As with our TREC 2006 system, only candidate answers that were extracted from the top 50 re-ranked documents were considered by downstream *Answer Selection* and *Answer Validation* modules.

Answer Retrieval and Extraction

In this section, we the three different answer retrieval strategies that CHAUCER-2 leverages in order provide answers to factoid questions.

Extracting Answers from the Factoid Database CHAUCER-2's first factoid Q/A strategy takes advantage of the large repository of factual information stored in its *Factoid Database* in order to find answers to a fixed set of question types. Under this approach, a series of heuristics are used to transform specific types of questions into database queries designed to retrieve specific information from the *Factoid Database.* For example, given a question like (Q282.2) *What is Pamuk's year of birth?*, heuristics employed by CHAUCER-2 will retrieve the BIRTH-YEAR field associated with a record with a NAME label of *Pamuk.* While we were encouraged by the precision of this approach, this strategy ultimately was limited in terms of the coverage and precision of the mapping heuristics we employed to convert questions into database queries. In future work, we plan to explore a multi-tiered classification approach – similar to the one we have employed for Answer Type Detection – in order to directly map between questions and individual fields stored in the *Factoid Database*.

Pattern-based Answer Extraction Previous versions of LCC's question-answering systems (Harabagiu *et al.* 2005; Hickl *et al.* 2006c) have successfully used libraries of handcrafted patterns in order to retrieve – and extract – candidate answers from collections of texts. Despite their promise (and their precision), pattern-based approaches have faced three significant challenges which have ultimately limited their recall. First, in order to be effective, pattern-based systems must include large libraries of patterns which account for a significant portion of the different types of questions that users will ask. Second, pattern-based systems also need to have access to accurate heuristics which will map different types of questions to the classes of patterns which can be used to extract answers. Finally, pattern-based systems need to be used in conjunction with high-recall document retrieval engines: if the relevant text snippets aren't retrieved, pattern-based systems will not be able to return answers. In order to counter this third challenge, CHAUCER-2 leverages a new index annotation framework which makes it possible to retrieve all of the text snippets matched by a pattern-based (or variabilized) query – without having to retrieve documents using a traditional keyword-based query. CHAUCER-2's index annotation framework (based on work first done by (Cafarella & Etzioni 2005) for an information extraction application) which makes it possible to extract all of the text passages matching an extraction pattern in a text collection without having to retrieve documents through an information retrieval engine. Following (Cafarella *et al.* 2005), we developed our own retrieval engine – which we refer to as the*neighborhood retrieval engine* – which can return short text snippets in response to variabilized queries. For ex-

ample, given a query like `TYPE PERSON NAME` such as `ProperNoun(Head(NP))`, our engine will return the set of entities marked as `TYPE PERSON NAME` which are followed by the sequence of the string such as and any proper noun which also heads an noun phrase (NP).

CHAUCER-2's *neighborhood retrieval engine* processes variables like `TYPE PERSON NAME` or `ProperNoun(Head(NP))` by returning every possible string in the corpus that has a matching type and that can be substituted for the variable and still satisfy the user's query. In order to retrieve the extensions of these variables quickly and without having to post-process documents, we again followed (Cafarella et al. 2005) in creating a new type of augmented inverted index, known as a *neighborhood index*, which allows for the processing of these queries with *O(k)* random disk seeks and *O(k)* disk reads, where *k* is defined as the number of non-variable terms in a query. In addition to keeping a list of the documents in which a term occurs – and a list of positions where the term occurs, the neighborhood index also stores a list of left-hand and right-hand neighbors at each position. The neighborhood contains the tokens token the left and right of the center token as well as any named entities and phrase chunks that end just before the token or start just after the token. Neighborhoods are additionally constrained to avoid crossing sentence boundaries.

Neighborhood indices are built by loading the documents from a normal Lucene index in order to produce a separate index just to represent neighborhoods. Most stop words are indexed because they can be important for certain query types, although queries involving stop words take much longer to execute than other queries. To reduce the size of the index, common words are stored in a dictionary and the index contains 1- or 2-byte pointer into the dictionary. Less common words are stored verbatim in the index. When determining which entities and phrase chunks are adjacent to a given token, some tokens are skipped. These tokens include articles, the word "who", quotation marks, and parenthesis. Skipping over these tokens dramatically increases the recall of some queries. These "noise" tokens are not stop words in the traditional sense; it is possibly to include these tokens in a query, but their presence in a document does not prevent neighborhoods from being found.

Entity-based Answer Extraction As with the TREC 2006 version of CHAUCER, CHAUCER-2's *entity-based* answer extraction strategy uses the distribution of named entities (recognized by LCC's CICEROLITE named entity recognition system) in order to identify candidate answers to individual questions. Under this approach, passages containing entity types associated with the question's expected answer type are first retrieved from the set of documents retrieved by the system. Candidate answers found within each passage are then extracted and re-ranked based on the distribution and density of question keywords discovered in each passage.

While traditional entity-based Q/A strategies have shown much promise in previous TREC QA evaluations (Harabagiu *et al.* 2005), they often retrieve many spurious answers which can greatly complicate the tasks of Answer Ranking and Answer Selection. In our TREC 2007 work, we hypothesized that if we could retrieve candidate answers not just based on the distribution of entity types – but in terms of specific conjunctions of semantic features extracted from a question – we could constrain the total number of candidate answers that are retrieved for a question without experiencing any reduction in overall precision.

In our experiments, we investigated how five different semantic features – based on the distribution of semantic dependencies in an candidate answer (as recognized by LCC's PropBank and NomBank parsers) – could be used in order to enhance the precision of a traditional entity-based answer extraction strategy. These five features included the presence of a semantic dependency found (1) between an entity in the answer (Ent_{ans}) corresponding to the question's expected answer type and a predicate (Pred_{ans}) corresponding to a the question's predicate answer type term, (2) between the Ent_{ans} and any other predicate in the candidate answer, (3) between the Pred_{ans} and any other argument in the candidate answer, (4) between an argument in the candidate answer (Arg_{ans}) corresponding to an argument from the question and any other predicate in the answer, and (5) between the Arg_{ans} and the Pred_{ans}. (A summary of the 10 different strategies are presented in Table 6.[3])

Strategy	Ent	Ent-Pred	Ent-$*$	$*$-Pred	Arg-$*$	Arg-Pred
1	✓	×	×	×	×	×
2	✓	✓	×	×	×	×
3	✓	×	✓	×	×	×
4	✓	×	×	✓	×	×
5	✓	×	×	×	✓	×
6	✓	×	×	×	×	✓
7	✓	×	✓	✓	×	×
8	✓	×	✓	×	✓	×
9	✓	×	×	✓	✓	×
10	✓	✓	✓	×	×	×

Table 6: Query Strategies used by CHAUCER-2's Entity-Based Q/A Strategy.

While Strategy 1 in (Table 6 corresponds to the default *entity-based* Q/A strategy, Strategies 2 through 10 represent contexts in which the retrieved candidate answers are subject to additional constraints. For example, Strategy 8 requires that all retrieved candidate answers must meet two conditions. First, any valid candidate answer must include an entity that corresponds to the expected answer type of the question that also participates in a predicate-argument relation with a predicate. In addition, the answer must also include an instance of an argument from the question which participates in a predicate-argument relationship with a predicate as well.

In our early work, we found that most of the query strategies listed in Table 6 returned few (if any) candidate answers for most questions; however, their precision (when parser

[3]We selected these 10 strategies to experiment with during our preparations for TREC 2007. We plan to explore the other possible combinations of features in future work.

errors where taken into account) in many cases approached 100%. In order to capitalize on these high-precision, low-recall strategies, we impelmented these 10 strategies as another cascade, which ranged from the most restrictive strategies (i.e. the ones which included the most constraints) to the least restrictive (i.e. Strategy 1, the traditional entity-based strategy). Although we considered candidate answers retrieved by all 10 strategies during *Answer Ranking* and *Answer Validation*, candidate answers were assigned a weight corresponding to the query strategy (or strategies) which was responsible for retrieving them; answers retrieved by more restrictive strategies received higher weights than those retrieved by less restrictive strategies.

Answer Ranking

Following *Answer Extraction*, CHAUCER uses a Maximum Entropy-based re-ranker (similar to (Ravichandran, Hovy, & Och 2003)) in order to compile answers from each of the six answer extraction strategies into a single ranked list. This re-ranker was trained on the top ten answers returned by each of CHAUCER's answer extraction strategies for each of the questions taken from the TREC 2004 and TREC 2005 datasets. (Answers were keyed automatically using "gold" answer patterns made available by the TREC organizers and other participating teams.) Five sets of features were used in this re-ranker: (1) the strategy used to extract the answer, (2) the EAT of the original question, (3) the entity type associated with the exact answer, (4) the redundancy of the answer across the top-ranked answers, and (5) the confidence assigned to the answer by each answer extraction strategy.

Answer Selection

Once a ranking of candidate answers is performed, the top 25 answers were then sent to an *Answer Selection* module which leverages LCC's state-of-the-art textual entailment system in order to identify the answer which best approximates the semantic content of the original question. Popularized by the recent PASCAL Recognizing Textual Entailment (RTE) Challenges (Dagan, Glickman, & Magnini 2005), textual entailment systems seek to identify whether the meaning of a *hypothesis* can be reasonably inferred from the meaning of a corresponding *text*. While the RTE Challenges have focused to-date only on the computation of entailment relationships between sentence-length texts and hypotheses, our recent work (Harabagiu & Hickl 2006) has shown that current systems for recognizing TE can be leveraged to accurately identify entailment relationships between questions and answers – or even questions and other questions.

CHAUCER uses the entailment system described in (Hickl *et al.* 2006b) in order to estimate the likelihood that a question entails either (1) a candidate answer extracted by one of CHAUCER's six answer extraction strategies or (2) a predictive question generated by the *Predictive Question Generation* module. Following (Harabagiu & Hickl 2006), we first filtered all candidate answers that were not entailed by the original questions. The remaining candidate answers (including any remaining predictive question-answer pairs) were re-ranked based on the entailment confidence output by the RTE system. The top-ranked answer was then returned as our submitted answer.

4. List Questions

This section describes the multiple strategies that CHAUCER-2 uses in order to provide answers to list questions. In order to maximize both precision and recall of the list answers CHAUCER-2 returns, we developed two distinct types of answer-finding strategies: (i) strategies that find all globally correct answers from an external knowledge source, then choose the supported answers that actually exist in the text, and (ii) strategies that find possible answers in the text and retain only those that pass some form of validation. The Type (i) strategies we have investigated in our TREC 2007 work include an (1) *Authoritative Source* strategy, a (2) *Wikipedia list* strategy, and a (3) *Lexicon* strategy. We only investigated one Type (ii) strategies this year, however: a *Web Count* strategy first introduced in (Hickl *et al.* 2006c).

Figure 2: Wikipedia Infobox for "St. Peter's Basilica"

Authoritative Source List Strategy Similar to the *Structured Data* strategy implemented for factoid questions, CHAUCER-2's *Authoritative Source* strategy uses the sources of semi-structured data stored in the *Factoid Database* in order to provide answers to list questions. Although the *Factoid Database* was used primarily used for answering factoid questions, some list fields (such as the

types of lists found on sites like *imdb.org* or stored in the HTML table "infoboxes" found on many *wikipedia.org* pages) were extracted heuristically and stored in the *Factoid Database* prior to the TREC 2007 evaluation. As with factoid questions, heuristics are used to map common question types to the particular fields (and sources) which would be most likely contain a correct answer. In addition, we found that the the the nominal ATT (as recognized by the *Answer Type Detection* module could often be used to identify a field which could contain a relevant set of answers. For example, given a question like (278.5) *What architects were involved in building St. Peter's?*, we found that searching the *Factoid Database* for the term *architect* returned a pointer to the "infobox" included on the *wikipedia.org* page for *St. Peter's Basilica* which mentions the four architects who worked on the basilica (e.g. *Donato Bramante, Antonio da Sangallo the Younger, Michelangelo, and Giacomo della Porta*. (See Figure 2 for an example of a "infobox".)

Wikipedia List and Table Strategy Our second strategy sought to leverage lists and tables mentioned on relevant *Wikipedia*[4] pages in order to identify candidate answers for list questions. Under this strategy, keywords extracted from both the question and the series target were used to retrieve a set of relevant pages from *Wikipedia*. Heuristics used to extract lists (and to "unroll" HTML) tables were then used – in conjunction with entity information available from CICERO-LITE in order to identify sets of multiple candidate answers. For example, the question (217.6) *What are titles of albums featuring Jay-Z?* has answers that can be found in the table on *Wikipedia*'s "Jay-Z_discography" page.[5]

Lexicon List Strategy In a third strategy, we used the collection of more than 800 different lexicons included in LCC's CICEROLITE in order to provide answers to list questions. As with the previous two strategies, heuristics were used again to map between selected types of question types (and or question keywords) and each of the lexicons available to CHAUCER-2.CHAUCER-2 utilizes the lexicon list strategy if three conditions are met: (1) a lexicon exists that matches the nominal ATT, (2) the nominal ATT is sufficiently far down the answer type hierarchy (by default, 2 nodes) from a coarse type, and (3) there is a proper noun (which becomes a mandatory keyword) after the nominal ATT in the question. This means that a questions like "*What Republican senators supported the nomination?*" and "*What persons has Krugman criticized in his op-ed columns?*" will not use the lexicon strategy (due to conditions (3) and (2), respectively), while a question like "*What musicals did Kurt Weill write?*" will.

Web Count List Strategy As with our TREC 2006 system, CHAUCER-2 also utilizes a method based on term frequency counts (obtained from search engines like *Google*) in order to determine how much of an associate there was between a candidate answer and both the series target and answer type term. These two scores were then combined in order to rank each individual candidate answer; answers above a threshold were included in the set of candidate answers considered by the system.

Strategy Selection As with factoid questions, we again cast the problem of selecting answers from multiple strategies as a cascade: list answers were considered first from the (1) *Authoritative Source* strategy, followed by answers from the (2) *Wikipedia List and Table* strategy, the (3) *Lexicon List* strategy, and (4) the *Web Count* strategy. Answers were added to the list until there were a maximum number of answers – or until there were no answers with a confidence level above a fixed threshold to return.

5. Answering "Other" Questions

In this section, we describe our approach to answering the "other" questions included with each question in the TREC 2007 QA Main Task.

As with our TREC 2006 submission, CHAUCER-2 begins the process of finding answers to "other" questions by first computing two types of automatic topic representations, including: (1) weighted lists of topic relevant terms known as *Topic Signatures* (Lin & Hovy 2000) (TS_1) and (2) a corresponding weighted list of topic relevant relations, known as *Enhanced Topic Signatures* (Harabagiu 2004) (TS_2). (As described in (Hickl *et al.* 2006c), both topic representations are computed from the top 100 documents retrieved from the AQUAINT-2/BLOG-06 corpus using keywords extracted from the series target – and all of the previous questions contained in the question series.)

Nugget Extraction

Once sets of TS_1 terms and TS_2 relations have been computed, CHAUCER-2 retrieves the top 500 documents from the AQUAINT-2/BLOG-06 corpus which contain at least one keyword from the series target. Passages are then extracted and ranked based on the top 25 most topical terms and relations. The top 500 passages retrieved using this method are then split into individual clauses using the sentence decomposition techniques introduced in (Hickl & Bensley 2007) and then were made available to the following four nugget extraction techniques.

"Web Words" Nugget Extraction Following an approach proposed by (Kaisser, Scheible, & Webber 2006), we used the top 50 most frequently-occurring non-stop words found in the first 100 pages retrieved from *Google* containing the series target in order to rank sentences retrieved from the AQUAINT-2/BLOG-06 corpus. Top-scoring sentences were then sent to an *Answer Selection* module to be combined with output from the other nugget extraction techniques.

Topic-Based Nugget Extraction Following work done by (Lacatusu *et al.* 2006) for question-focused summarization, we used weights associated with TS_1 terms and TS_2 relations to compute a composite *topic score* for each sentence in the set of documents retrieved for a target. Sentences were re-ranked based on their *topic score* before being submitted to the *Answer Selection* module.

[4] http://www.wikipedia.org

[5] http://en.wikipedia.org/wiki/Jay-Z_discography

Soft Pattern-Based Nugget Extraction As with our TREC 2006 submission, we again experimented with using the probabilistic soft matching techniques first described in (Cui, Kan, & Chua 2004) in order to identify additional patterns that could be used to extract nuggets for a particular target type. we followed (Cui, Kan, & Chua 2004) in developing a bigram soft pattern model in order to identify potential matches between a set of training sentences and each of the sentences extracted for a particular target. Training sentences were derived for each target type from two different sources: (1) the collection of "gold" nuggets identified for the TREC 2005 "other questions" and a collection of 5,000 biographies, descriptions, and encyclopedia articles that were downloaded from the collection of "authoritative sources" used to populate CHAUCER-2's factoid database.

Headline Extraction In addition to nuggets retrieved using the previous three strategies, CHAUCER-2 also retrieves all of the document headlines which contain both the series target and at least one TS_1 term or TS_2 relation. While not every series target appeared in a headline of a document contained in the AQUAINT-2 collection, we found that headlines often contained a succinct, topical statement that was not unlike the "gold standard" nuggets reported as the keys for "other" questions. Since headlines appeared to provide consistently good information for "other" questions, they were not submitted to the *Answer Selection* module, but appended to the top of each submitted set of nuggets.

Answer Combination

In a departure from the content modeling approach introduced in (Hickl *et al.* 2006c), we used a simple combination method to combine (and rank) candidate nuggets for submission. Following work done by (Lacatusu *et al.* 2006) for the DUC multi-document summarization evaluations, candidate nuggets were assigned a composite score based on the density of TS_1 terms and TS_2 relations as well as the individual rank that they were assigned by each individual nugget extraction technique. All nuggets which received a score above a fixed threshold were returned as part of our official submission.

6. Evaluation Results

Table 7 presents a summary of CHAUCER-2's performance on the TREC 2007 QA Main Task.

Task	Evaluation Metric	CHAUCER-2
Factoid Q/A	Accuracy	56.1%
List Q/A	Fβ1	32.4%
"Other" Q/A	Fβ3	26.1%
Series Score	Aggregate	35.8%

Table 7: Summary of TREC 2007 QA Main Track Results.

A detailed breakdown of the results from the Factoid Q/A task is presented in Table 8.

TREC 2007 marked the first year where we made a concentrated effort to develop a coherent strategy for answering list questions. We believe our results to be encouraging: our TREC 2007 results more than doubled our TREC 2006 results in terms of recall, precision, and F-measure (Fβ1).

Judgment	Percent
Wrong	37.5%
Unsupported	2.7%
Inexact	4.7%
Locally Correct	1.2%
Globally Right	53.8%

Table 8: TREC 2007 Factoid Q/A Results

Metric	TREC 2007
Recall	0.361
Precision	0.412
F(β=1)	0.324

Table 9: TREC 2007 List Q/A Results

Finally, Table 10 shows our precision, recall, and F-Score for "other" questions.

Metric	TREC 2007
Recall	0.288
Precision	0.2501
F(β=3)	0.261

Table 10: TREC 2006 Other Q/A Results

6. Conclusions

This paper describes CHAUCER-2, the most recent version of Language Computer Corporation's CHAUCER line of automatic question-answering systems. Developed for the 2007 TREC QA Track Main Task, CHAUCER-2 was designed to explore how a new, semantically-rich framework for information retrieval could be used to boost the overall performance of the answer extraction and answer selection components of an end-to-end question-answering system.

First, unlike the keyword-based retrieval systems used by LCC's previous Q/A systems (Harabagiu *et al.* 2005; Hickl *et al.* 2006c), CHAUCER-2 employed a novel indexing and retrieval engine which supported a wide range of semantically-rich queries, including queries based on semantic types recognized by LCC's CICEROLITE named entity recognition system as well as semantic dependencies identified by LCC's PropBank-, NomBank-, and FrameNet-based semantic parsers. We found that support for these new types of queries dramatically enhanced the performance of the retrieval components used in CHAUCER-2 while significantly reducing the number of candidate answers that had to be considered by CHAUCER-2s *Answer Ranking* and *Answer Validation* modules.

In addition to supporting multiple query types, CHAUCER-2 also leveraged a variant of the Bindings Engine (BE) first proposed by (Cafarella & Etzioni 2005; Cafarella *et al.* 2005) in order to retrieve of all of the text snippets matched by a query without having to retrieve documents using a keyword-based query. We found that use of this framework greatly both enhanced the efficiency and the recall of traditional pattern-based approaches to Q/A.

Finally, CHAUCER-2 incorporated a new, multi-tiered Answer Type Detection (ATD) module which reduced the

number of expected answer types (EATs) considered by the system for factoid or list questions, while maintaining the same high levels of precision exhibited by previous systems.

Acknowledgments

The authors would like to thank Sanda Harabagiu, Paul Aarseth, John Lehmann, and Luke Nezda for their assistance with this work. This material is based upon work funded in whole or in part by the U.S. Government and any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Government.

References

Cafarella, M., and Etzioni, O. 2005. A Search Engine for Natural Language Applications. In *Proceedings of the Fourteenth World Wide Web conference (WWW 2005)*.

Cafarella, M.; Downey, D.; Soderland, S.; and Etzioni, O. 2005. KnowItNow: Fast, Scalable, Information Extraction from the Web. In *Proceedings of EMNLP-2005*.

Chakrabarti, S.; Krishnan, V.; and Das, S. 2005. Enhanced answer type inference from questions using sequential models. In *Proceedings of EMNLP*.

Cui, H.; Kan, M.-Y.; and Chua, T.-S. 2004. Unsupervised Learning of Soft Patterns for Definitional Question Answering. In *Proceedings of the Thirteenth World Wide Web conference (WWW 2004)*, 90–99.

Dagan, I.; Glickman, O.; and Magnini, B. 2005. The pascal recognizing textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop*.

Harabagiu, S., and Hickl, A. 2006. Methods for Using Textual Entailment in Open-Domain Question Answering. In *Proceedings of COLING-ACL*.

Harabagiu, S.; Moldovan, D.; Clark, C.; Bowden, M.; Hickl, A.; and Wang, P. 2005. Employing Two Question Answering Systems in TREC 2005. In *Proceedings of the Fourteenth Text REtrieval Conference*.

Harabagiu, S.; Hickl, A.; and Lacatusu, F. 2006. Negation, Contrast and Contradiction in Text Processing. In *Proceedings of AAAI-06*.

Harabagiu, S. 2004. Incremental Topic Representations. In *Proceedings of the 20th COLING Conference*.

Hickl, A., and Bensley, J. 2007. A Discourse Commitment-based Framework for Recognizing Textual Entailment. In *Proceedings of the ACL 2007 Workshop on Paraphrasing and Textual Entailment*.

Hickl, A., and Harabagiu, S. 2007. Machine Reading through Textual and Knowledge Entailment. In *Proceedings of the 2007 AAAI Spring Symposium on Machine Reading.*

Hickl, A.; Wang, P.; Lehamnn, J.; and Harabagiu, S. 2006a. Ferret: Interactive Question-Answering for Real-World Research Environments. In *Proceedings of the 2006 COLING-ACL Interactive Presentations Session*.

Hickl, A.; Williams, J.; Bensley, J.; Roberts, K.; Rink, B.; and Shi, Y. 2006b. Recognizing Textual Entailment with LCC's Groundhog System. In *Proceedings of the Second PASCAL Challenges Workshop (to appear)*.

Hickl, A.; Williams, J.; Bensley, J.; Roberts, K.; Shi, Y.; and Rink, B. 2006c. Qusetion Answering with LCC's Chaucer at TREC 2006. In *Proceedings of the Fifteenth Text REtrieval Conference*.

Kaisser, M.; Scheible, S.; and Webber, B. 2006. Experiments at the University of Edinburgh for the TREC 2006 QA Track. In *Proceedings of the Fifteenth Text REtrieval Conference*.

Lacatusu, F.; Hickl, A.; Roberts, K.; Shi, Y.; Bensley, J.; Rink, B.; Wang, P.; and Taylor, L. 2006. Lcc's gistexter at duc 2006: Multi-strategy multi-document summarization. In *Proceedings of DUC 2006*.

Lehmann, J.; Aarseth, P.; Nezda, L.; Deligonul, M.; and Hickl, A. 2005. TASER: A Temporal and Spatial Expression Recognition and Normalization System. In *Proceedings of the 2005 Automatic Content Extraction Conference*.

Li, X., and Roth, D. 2002. Learning question classifiers. In *Proc. the International Conference on Computational Linguistics (COLING)*.

Lin, C.-Y., and Hovy, E. 2000. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th conference on Computational linguistics*, 495–501. Morristown, NJ, USA: Association for Computational Linguistics.

Ravichandran, D.; Hovy, E.; and Och, F. 2003. Statistical qa - classifier vs re-ranker: What's the difference? In *Proceedings of the ACL Workshop on Multilingual Summarization and Question Answering*.