Question Answering with *Lydia* (TREC 2005 QA track)

Jae Hong Kil, Levon Lloyd, and Steven Skiena

Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400
{jkil, lloyd, skiena}@cs.sunysb.edu

1 Introduction

The goal of our participation in TREC 2005 was to determine how effectively our entity recognition/text analysis system, *Lydia* (*http://www.textmap.com*) [1–3] could be adapted to question answering. Indeed, our entire QA subsystem consists of only about 2000 additional lines of Perl code. *Lydia* detects every named entity mentioned in the AQUAINT corpus, and keeps a variety of information on named entities and documents in a relational database. We can collect candidate answers by means of information kept in the database. To produce a response for the main task or a ranked list of documents for the document ranking task, we rank the collected candidate answers or documents using syntactic and statistical analyses.

A significant distinction from other question answering systems [4–6] presented earlier at TREC is that we do *not* use web sources such as *Wikipedia and Google* to generate candidate answers or answers. Rather, we only use syntactic and statistical features of the test set of questions and corpus provided. Our approach is independent of other sources, and finds answers from the text provided.

We describe the design of *Lydia* and associated algorithms in Section 2, and focus on the design and algorithms of the QA system in Section 3. We then analyze the performance of the QA system in Section 4, and conclude this paper with discussion on future directions in Section 5.

2 The *Lydia* System

Lydia is designed for high-speed analysis of online text, and it analyzes thousands of curated text feeds daily. *Lydia* is capable of retrieving a daily newspaper like *The New York Times* and then analyzing the resulting stream of text in under one minute of computer time.

These design criteria force us to abandon certain standard techniques from natural language processing as too slow, most notably grammar-based parsing techniques. Instead, we use part of speech tagging [7] to augment the performance of special-purpose pattern matchers to recognize names, places, and other proper nouns, as well as related patterns of interest.

A block diagram of the *Lydia* processing pipeline appears in Figure 1.

2.1 Named Entity Recognition

Fundamental to *Lydia* is a problem referred to as *named entity recognition* within the natural language processing literature, where one seeks to detect every named entity mentioned in a document. The most important phases of our system for named entity recognition are as follows:

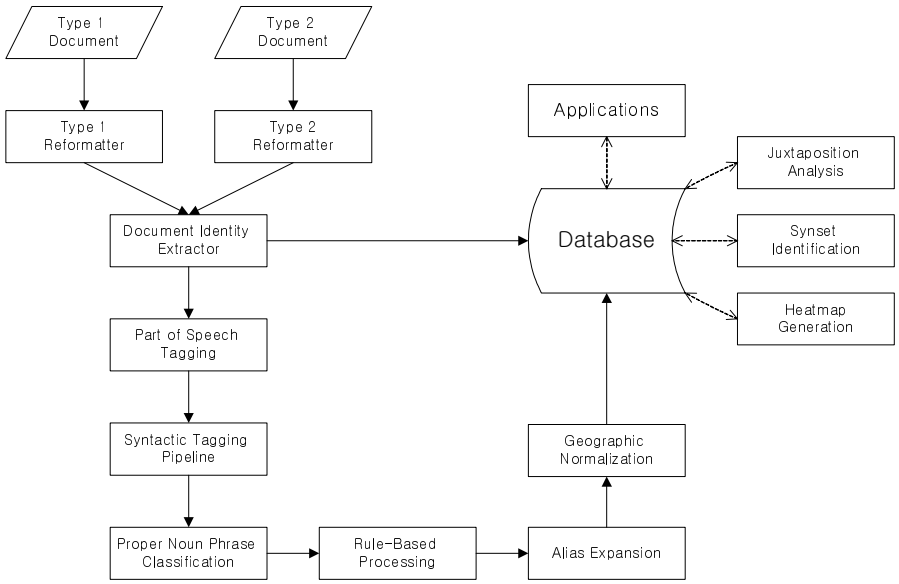

Fig. 1. Block Diagram of the *Lydia* Processing Pipeline

Part of Speech Tagging To extract proper noun phrases from text, we tag each input word with an appropriate part of speech tag (noun, verb, adjective, etc.) on the basis of statistical predication and local rules. We employ Brill's popular part of speech (POS) tagger in our analysis.

Syntactic Tagging In this phase of the pipeline, we use regular-expression patterns implemented in Perl to markup certain classes of important text features such as *dates*, *numbers*, and *unit-tagged quantities*.

Proper Noun Phrase Classification Each proper noun phrase in a text belongs to some semantic class, such as *person*, *city*, or *company*. We first attempt to classify each entity by looking it up in a series of gazetteers. If that fails, then we employ a Bayesian classifier[8].

Rule-Based Processing Compound entities are difficult to handle correctly. For example, the entity name *State University of New York, Stony Brook* spans both a comma and an uncapitalized word that is not a proper noun. By comparison, *China, Japan, and Korea* refers to three separate entities. Our solution is to implement a small set (~ 60) of hand-crafted rules to properly handle such exceptions.

Alias Expansion A single entity is often described by several different proper noun phrases, e.g. *President Kennedy*, *John Kennedy*, *John F. Kennedy*, and *JFK*, even in the same document. We identify two common classes of aliasing, *suffix aliasing* and *company aliasing*, and take appropriate steps to unify such representations into a single set.

Geographic Normalization Geographic names can be ambiguous. For example, *Albany* is both the capital of New York State and a similarly-sized city in Georgia. We use a geographic normalization routine that identifies where places are mentioned, resolves any ambiguity using

population and locational information, and replaces the name with a normalized, unambiguous representation.

2.2 Juxtaposition Analysis

A primary goal of *Lydia* is to measure how entities relate to each other. Given a pair of entities in the database, we seek to assign a score to the *juxtapositionness* of them and then for any given entity we can find the other entities that scored the highest with it.

To determine the significance of a juxtaposition, we bound the probability that two entities co-occur in the number of articles that they co-occur in if occurrences where generated by a random process. To estimate this probability we use a Chernoff Bound:

$$P(X > (1+\delta)E[X]) \leq (\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}})^{E[X]}$$

where δ measures how far above the expected value the random variable is. If we set $(1+\delta)E[X] = F =$ number of co-occurrences, and consider X as the number of randomized juxtapositions, we can bound the probability that we observe at least F juxtapositions by calculating

$$P(X > F) \leq (\frac{e^{\frac{F}{E[X]}-1}}{(\frac{F}{E[X]})^{(\frac{F}{E[X]})}})^{E[X]}$$

where $E[X] = \frac{n_a n_b}{N}, N =$ number of sentences in the corpus, $n_a =$ number of occurrences of entity a, and $n_b =$ number of occurrences of entity b, as the juxtaposition score for a pair of entities. We display $-\log$ of this probability for numerical stability and ranking.

3 Question Answering and Document Ranking

Our QA system is designed to answer *factoid, list*, and *other* questions, and to rank the relevance of documents for each question. The QA system performs question answering and document ranking through three partially overlapped flows by means of information kept in the database of *Lydia* and tagged AQUAINT corpus which are acquired by running the AQUAINT corpus through the *Lydia* pipeline.

A block diagram of the QA system appears in Figure 2.

3.1 Factoid and List Questions

To produce an answer for a *factoid* and answers for a *list* question, our QA system processes five phases following flowline (1) in Figure 2. The five phases are as follows:

Question Preprocessing To analyze question types, we part of speech tag the test set of questions provided. We employ Brill's part of speech (POS) tagger in the questions.

ex) Where is Port Arthur? ⇒ Where/WRB is/VBZ Port/NNP Arthur/NNP ?/.

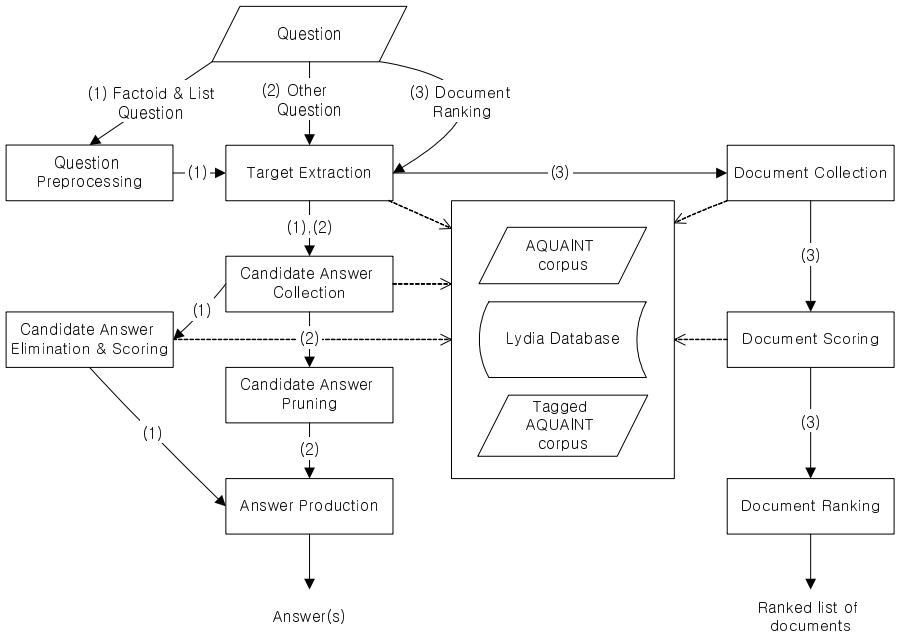

Fig. 2. Block Diagram of the Question Answering System

Target Extraction The *Lydia* database keeps track of every named entity recognized. To use information in the database, we extract a target, for example, "Kim Jong Il". When *Lydia* does not recognize a target, we extract one as follows:

- *Valid Form Extraction* – When a target is not in a valid form such as a plural noun, we use WordNet [9] to obtain a valid form of the target.

- *Partial Extraction* – *Lydia* recognizes proper nouns and various types of numerical nouns as named entities. However, *Lydia* does not recognize a phrase or a compound word such as "Russian submarine Kursk sinks" as a named entity. In that case, we extract a part of the target, for example, "Kursk" which is recognized by *Lydia*.

- *Synonym Extraction* – If a target and any part of the target are not recognized by *Lydia*, we acquire synonyms of the target using WordNet, and then choose a synonym which is recognized by *Lydia* as a target.

Candidate Answer Collection We identify which sentences in the documents contain the target by querying the *Lydia* database, as well as all the entities which are in these sentences. Since we only identify the sentences where the target is, it takes comparatively less time to collect candidate answers than with whole documents. In addition, it enables us to more specifically identify which part in the text provides an answer.

Candidate Answer Elimination and Scoring To rank the candidate answers, we use three types of analysis algorithms.

- *Juxtaposition Analysis* – We obtain the juxtaposition score between a target and each candidate answer querying the *Lydia* database. More strongly associated terms are likely to

	Type	Question
1	(What|Which) + (the|an|a|NN of|) + (NN|JJ + NN)	Which countries expressed regret about the loss?
2	(List|Name|. . .) + (the|an|a|NN of|) + (NN|JJ + NN)	Name players on the French team.
3	How + (NN|JJ|RB)	How many students were wounded?
4	(Who|Whose|Whom|When|Where|Why)	When was Enrico Fermi born?
5	(How|What|Which)+ VB	What is Hong Kong's population?

Table 1. Five Question Types with Example Questions

	CARDINAL	COMPANY	COUNTRY	DATE	DISEASE	PERSON	TIMEPERIOD	VOLUME	WEBSITE
WHO	N/A	2	2	N/A	N/A	10	N/A	N/A	N/A
WHOSE	N/A	2	2	N/A	N/A	10	N/A	N/A	N/A
WHOM	N/A	2	2	N/A	N/A	10	N/A	N/A	N/A
WHEN	10	N/A	N/A	10	N/A	N/A	N/A	N/A	N/A
WHERE	N/A	2	10	N/A	1	1	N/A	N/A	1
WHY	N/A	1	1	N/A	5	1	N/A	N/A	N/A
WHAT	2	10	2	2	10	5	2	2	10
WHICH	2	10	2	2	10	5	2	2	10
HOW	N/A	1	1	N/A	5	1	N/A	N/A	N/A

Table 2. Relevance Scores of Nine Interrogatives and Selected Classes

represent answers to commonly asked questions.

– *Question Term Analysis* – To analyze question terms in a question, we first eliminate interrogative phrases and stop words [10]. We then obtain a valid form of each non-trivial question term and its synonyms. We weight each candidate answer identified in a sentence with both the target and a non-trivial question term/its synonym using the following equation:

$$weight = C \times \frac{n_q}{n_t}$$

where C is a constant(≈ 100), n_q = number of non-trivial question term or its synonym, and n_t = number of total non-trivial question terms.

– *Question Type Analysis* – Our methods for eliminating and scoring candidate answers vary with question types. Table 1 shows five question types and their example questions.

- For question types 1 and 2, the class[1] of an answer is likely to be the same as the noun followed by a verb. For example, "country" is the valid form of "countries" in the example question 1 in Table 1, thus we eliminate candidate answers whose class is not the same with the noun, *country*. If the noun followed by a verb does not match with any class in *Lydia*, we obtain hypernyms of the noun using WordNet. For example, the noun, "players" in the example of question type 2 in Table 1 does not match with any class. However, a hypernym of "players" is *person*, and the class of the answer for the question is likely to be *person*, so we eliminate candidates answers whose class is not *person*.

[1] Currently, *Lydia* classifies named entities into about 60 semantic classes, e.g. *cardinal, country, date and university*. We are still developing gazetteer-based algorithm and Bayesian classifier to classify entities more accurately and specifically. Hence, the number of semantic class is steadily increasing.

- The answers for question type 3 are obviously *numerical*. Therefore, we eliminate candidate answers whose class is not numerical such as *company, country, disease or person.*
- An interrogative or interrogative phrase does not provide a reliable clue to recognize the class of an answer for question types 4 and 5, thus we use a set of lists, "*interrogative, class, and relevance score*" for all interrogatives, which are *who, whose, whom, when, where, why, what, which, and how.* Table 2 shows the relevance scores of the nine interrogatives and the selected classes. For example, the interrogative *when* is on five lists as follows: WHEN CARDINAL 10, WHEN DATE 10, WHEN MONTH 2, WHEN DAY 2, WHEN TIME 2. This means that when the interrogative of a question is *when*, the class of an answer can be only one of the five classes, *cardinal, date, month, day, and time*, and the *relevance score* for each candidate answer whose class is *cardinal, date, month, day, or time* is 10, 10, 2, 2, or 2, respectively.

Answer Production To produce an answer for a *factoid* question or answers for a *list* question, we rank candidate answers as follows:

– *Candidate Answer Ranking* – We score each candidate answer depending on question type, for question types 1, 2 and 3:

$$score = juxtaposition\ score \times weight$$

for question types 4 and 5:

$$score = juxtaposition\ score \times weight \times relevance\ score^{C}$$

where C is a constant(≈ 2). We then sort all the candidate answers in descending order of score.

– *Answer Selection* – An answer for a *factoid* question is the entity whose score is the highest, and answers for a *list* question are the entities whose scores are higher than a threshold.

3.2 Other Question

To produce answers for an *other* question, our QA system processes the four phases following flowline (2) in Figure 2. Answering an *other* question does not require the *question preprocessing* phase since an *other* question in the test set of questions provided is not in sentence form, but merely a word, "Other". The *candidate answer pruning* phase plays a role in narrowing down candidate answers as the *candidate answer elimination and scoring* phase does for a *factoid* and a *list* question.

The four phases are as follows:

Target Extraction Basically, the algorithms for an *other* question are the same with the ones for a *factoid* and a *list* question. In addition, we employ a set of *first name-nickname pairs* [11] to expand the target using the *first name equivalence*, since we need to obtain the largest set of sentences which might contain definitions of the target. For example, since *Jim* is a nickname of *James*, *Jim Inhofe* and *James Inhofe* indicate the same person. Assuming that two consecutive words starting with a capital in a target represent a person name, we search the first word in the set of first name-nickname pairs. If found, we use both the original target and the new target replaced the first word by its pair as targets.

Candidate Answer Collection We obtain the documents which contain a target or targets by querying the *Lydia* database, and collect all the sentences which contain the target or targets.

Candidate Answer Pruning Though every sentence contains information on a target or targets, we need to prune sentences which are duplicated or not directly related to the target or targets. Our pruning algorithms are as follows:

- *Sentence Preprocessing* – At first, we remove meaningless snippets such as newspaper title, publication date and place in each sentence. We then eliminate too short sentences which barely convey significant meanings, and sentences whose subjects are the first-person or the second-person, namely "I", "We" or "You" since they are highly likely to only contain a subjective opinion of the subject. In addition, fully or partially duplicated sentences are eliminated.

- *Sentence Scoring* – We assign the same initial score to each sentence, and weight or deweight each sentence. We weight a sentence which fully contains the target more than one with the target in part, and a sentence which contains the possible syntactic pattern of a definition, *Target + (is|was|who|which|that) or Target(s|es) + (are|were|who|which|that)*. On the other hand, we deweight a comparatively long or short sentence (the threshold is determined to be 100 due to the allowance of characters for each correct nugget [12, 13]) as well as those which contain unrelated words such as "say", "ask", "report", "If", "Unless", interrogatives, and subjective pronouns. We also deweight a sentence which contains too many non-trivial words or proper nouns compared to other types of words since the sentence is likely to be just an enumeration of nouns such as names and places.

Answer Production We sort all the candidate answers in descending order, and then select sentences whose scores are higher than a threshold.

3.3 Document Ranking

Since our QA system does not rank documents for the purpose of answering three types of questions, we employ a function of *length of a document, number of occurrences of a target* and *number of occurrences of marked (tagged) terms*, or *length of a document* and *number of occurrences of marked (tagged) terms* in the *document scoring* phase.

The QA system processes four phases following flowline (3) in Figure 2. The four phases described in two sections, *target extraction, document collection, document scoring*, and *document ranking* are as follows:

Target Extraction and Document Collection We extract a target with the same algorithms for a *factoid* and a *list* question, and then identify all the documents which contain the target or synonyms of the target querying the *Lydia* database.

Document Scoring and Ranking To obtain a ranked list of 1000 documents for each question, we follow the three steps:

– *Documents with the target* – We score each document with the function:

$$score = C_1 \times l_d + C_2 \times n_t + C_3 \times n_m$$

where C_1, C_2 and C_3 are constants, l_d = length of a document, n_t = number of occurrences of a target, and n_m = number of occurrences of marked (tagged) terms. We then sort the documents in descending order.

– *Documents with synonyms of the target* – The scoring function is the same with the one for documents with the target. After scoring each document, we sort the documents in descending order, and scale down the scores of the sorted documents to rank them lower than the lowest ranked document which contains the target.

– *Completing the ranking* – To ensure we submit 1000 documents for each query, we attempt to identify the top 1000 documents for a "null query", and use these to make up the balance. We score all the documents in the AQUAINT corpus with the function:

$$score = C_1 \times l_d + C_2 \times n_m$$

where C_1 and C_2 are constants, l_d = length of the document, and n_m = number of occurrences of marked (tagged) terms. We then sort the documents in descending order, and generate a list of 1000 documents with the highest score. This procedure is done once since it is independent of questions. Finally, we fill the balance from the 1000 documents if the number of documents which contain the target and synonyms of the target is less than 1000.

4 Performance Analysis

We submitted three runs for the main task and the document ranking task of TREC 2005 QA track: SUNYSB05qa1, SUNYSB05qa2 and SUNYSB05qa3. These three runs only differ in settings for answer production. The setting of SUNYSB05qa2 is less conservative than the one of SUNYSB05qa1, and the setting of SUNYSB05qa3 is more conservative than the one of SUNYSB05qa1. According to the final results, our QA system for *list* questions and *other* questions performs better when settings are less conservative while for *factoid* questions it produces better performance when settings are more conservative.

Table 3 shows performance of our three runs and the median scores of 71 runs submitted to TREC 2005 QA track. The scores of our runs for *list* questions and *other* questions are greater than the median scores of 71 runs, but the score for *factoid* questions is less than the one of 71 runs. The final scores of our runs are almost the same with the median final score of 71 runs.

	Factoid Question	List Question	Other Question	Final Score
SUNYSB05qa1	0.102	0.066	0.194	0.120
SUNYSB05qa2	0.105	0.064	0.196	0.121
SUNYSB05qa3	0.122	0.059	0.179	0.123
Median Score	0.152	0.053	0.156	0.123

Table 3. Performance of SUNYSB Runs in TREC 2005

Table 3 shows that performance of our QA system for *list* questions and *other* questions is better than the one for *factoid* questions in comparision of performance of other runs submitted to TREC 2005 QA track. This results from the following features of our QA system. It chooses answers using syntactic and statistical features of the test set of questions and corpus provided. The probability of selecting a single correct answer out of candidate answers for a *factoid* question is very low. On the other hand, the probability of selecting *some* correct answers for a *list* question and an *other* question is relatively high now that it is likely that collected candidate answers include some of the correct answers.

In general, the accuracy of extracting a proper target fairly affects performance of the QA system. For example, the QA system performs well for a target "Sammy Sosa" since it can properly extract the target and collect candidate answers based on the target. However, it does not perform well for complicated targets like "Plane clips cable wires in Italian resort" or "1998 Baseball World Series." In addition, in case that a correct answer is a non-named entity, performance of the QA system is not satisfactory since *Lydia* which the QA system essentially depends on mainly detects named entities in text.

5 Conclusions and Future Work

We have presented the design of the *Lydia* question answering, along with analyzing its performance.

We are continuing to improve *Lydia*, particularly the entity recognition algorithms, entity classification, and geographic normalization. Since main phases of the QA system such as target extraction, candidate answer collection, candidate answer elimination and scoring are essentially based on *Lydia*, improvement in *Lydia* will enhance the performance of the QA system.

Future directions of work on improving the QA system include exploring the use of a system which automatically extracts relations between entities by means of analyzing syntactic and semantic patterns of verbs.

References

1. L. Lloyd, D. Kechagias, and S. Skiena. Lydia: A system for large-scale news analysis. In *Proc. 12th Symp. of String Processing and Information Retrieval (SPIRE '05)*, Buenos Aires, Argentina, November 2-4 2005.
2. L. Lloyd, A. Mehler, and S. Skiena. Identifying synonymous names in large news corpra. in preparation, 2005.
3. Y. Bao, X. Li, A. Mehler, Y. Wang, and S. Skiena. Spatial analysis of news sources. in preparation, 2005.
4. David Ahn, Valentin Jijkoun, Gilad Mishne, Karin Muller, Maarten de Rijke, and Stefan Schlobach. Using wikipedia at the trec qa track, 2004.
5. Dmitri Roussinov, Jose Antonio Robles-Flores, and Yin Ding. Experiments with web qa system and trec2004 questions, 2004.
6. Lide Wu, Xuanjing Huang, Lan You, Zhushuo Zhang, Xin Li, and Yaqian Zhou. Fduqa on trec2004 qa track, 2004.
7. E. Brill. Some advances in rule-based part of speech tagging. In *Proccedings of the Twelfth National Conference on Artificial Intelligence*, 1994.
8. T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
9. Wordnet. http://wordnet.princeton.edu/.
10. Dvl/verity stop word list. http://dvl.dtic.mil/stop_list.html.
11. Nicknames and their possible full names. http://www.veritasinfo.com/page7.html.
12. Ellen M. Voorhees. Overview of the trec 2003 question answering track. In *Proceedings of the Twelfth Text REtreival Conference (TREC 2003)*, 2004.
13. Ellen M. Voorhees. Overview of the trec 2004 question answering track. In *Proceedings of the Thirteenth Text REtreival Conference (TREC 2004)*, 2005.