Learning to Answer Biomedical Questions: OAQA at BioASQ 4B

Zi Yang Yue Zhou Eric Nyberg
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
{ziy, ehn}@cs.cmu.edu

Abstract

This paper describes the OAQA system evaluated in the BioASQ 4B Question Answering track. The system extends the Yang et al. (2015) system and integrates additional biomedical and general-purpose NLP annotators, machine learning modules for search result scoring, collective answer reranking, and yes/no answer prediction. We first present the overall architecture of the system, and then focus on describing the main extensions to the Yang et al. (2015) approach. Before the official evaluation, we used the development dataset (excluding the 3B Batch 5 subset) for training. We present initial evaluation results on a subset of the development data set to demonstrate the effectiveness of the proposed new methods, and focus on performance analysis of yes/no question answering.

1 Introduction

The BioASQ QA challenge (Tsatsaronis et al., 2015) evaluates automatic question answering technologies and systems in the biomedical domain. It consists of two phases: in Phase A, the task requires to retrieve relevant document, snippets, concepts, and triples given a natural language question, and evaluates the retrieval results in terms of mean average precision (MAP); in Phase B, the task requires to generate ideal answers for the questions, which are evaluated using accuracy and mean reciprocal rank (MRR), as well as exact answers, which are evaluated based on manual judgment. The OAQA team participated in Batches 3, 4, and 5 of BioASQ 4B, in the categories of document, snippet, and concept retrieval, factoid, list and yes/no question answering (exact answer generation). The source code of the participating system can be downloaded from our GitHub repository[1].

We follow the same general hypothesis expressed in Ferrucci et al. (2009) and Yang et al. (2015), specifically that informatics challenges like BioASQ are best met through careful design of a flexible and extensible architecture, coupled with continuous, incremental experimentation and optimization over various combinations of existing state-of-the-art components, rather than relying on a single "magic" component or single component combination. This year, the number of labeled questions in the development set has grown to 1,307 (up from 810 in last year's dataset), which allows further exploration of a) the potential of supervised learning methods, and b) the effectiveness of various biomedical NLP tools in various phases of the system, from relevant concept and document retrieval to snippet extraction, and from answer text identification to answer prediction.

First, we use TmTool[2] (Wei et al., 2016), in addition to MetaMap[3], to identify possible biomedical named entities, especially out-of-vocabulary concepts. We also extract frequent multi-word terms from relevant snippets (Frantzi et al., 2000) to further improve the recall of concept and candidate answer text extraction. Second, we propose a supervised learning method to rerank the answer candidates for factoid and list questions based on the relation between each candidate answer and other candidate answers, which we refer to as collective reranking in this paper. Third, we implement a yes/no question answering pipeline combining various heuristics, e.g. negation words, sentiment of the statements, the biomedical con-

[1] https://github.com/oaqa/bioasq
[2] http://www.ncbi.nlm.nih.gov/CBBresearch/Lu/Demo/tmTools/
[3] http://metamap.nlm.nih.gov/

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 23–37,
Berlin, Germany, August 7-12, 2016.

cepts mentioned in the relevant snippets that belong to the same concept type, and question inversion (Kanayama et al., 2012). Finally, we introduce a unified classification interface for judging the relevance of each retrieved concept, document, and snippet, which can combine the relevant scores evidenced by various sources, e.g. retrieval scores using different queries and indexes.

This paper describes the system that was evaluated in the BioASQ 4B challenge. We first review the system architecture and the approaches used in Yang et al. (2015) in Section 2, and then we focus on describing each individual component for BioASQ 4B in Sections 3 to 6. Before the official evaluation, we trained the system using the development dataset excluding the 3B Batch 5 subset; we evaluate the proposed approach using the held-out 3B Batch 5 subset. Section 7 presents the results, which illustrate the effectiveness of the proposed methods, and Section 8 presents a manual error analysis of the proposed yes/no QA method and highlight the challenges of biomedical yes/no QA problem. We conclude and present future work in Section 9.

2 Overview of Yang et al. (2015) System

In this section, we briefly describe the architecture of the Yang et al. (2015) system, which provided the baseline for the system evaluated here. Further detail can be found in the full paper.

The Yang et al. (2015) system uses the UIMA ECD/CSE framework[4] (Garduno et al., 2013; Yang et al., 2013) with a YAML[5]-based language to support formal, declarative descriptors for the space of system and component configurations to be explored during system optimization. The system employs a three-layered architecture. The first layer BaseQA[6] is designed for domain-independent QA components, and includes the basic input/output definition of a QA pipeline, intermediate data objects, QA evaluation components, and data processing components. In the second layer, we implemented biomedical resources that can be used in any biomedical QA task (outside the context of BioASQ). A few BioASQ-specific components were integrated in the third design layer; for example, GoPubMed services are only hosted for the purpose of the BioASQ challenge. Tools and resources that are shared by multiple components are defined as `providers`, including NLP parsers, concept identification modules, synonym expansion modules, classifiers, etc.

Resources. The Yang et al. (2015) system uses LingPipe and ClearNLP[7] (Choi and Palmer, 2011) to parse the questions and relevant snippets using models applicable to generic English texts as well as biomedical texts, e.g. the parser models trained on the CRAFT treebank (Verspoor et al., 2012). It uses the named entity recognition (NER) module from LingPipe[8] trained on the GENIA corpus (Kim et al., 2003) and MetaMap annotation component (Aronson, 2001) to identify the biomedical concepts, and further uses UMLS Terminology Services (UTS)[9] to identify concepts and retrieve synonyms. It uses the official GoPubMed services for concept retrieval, and a local Lucene[10] index for document retrieval and snippet retrieval. LibLinear[11] (Fan et al., 2008) is used to train classifiers to predict answer types to the questions and estimate the relevance scores of candidate answers.

Answer Type Prediction. To identify the gold standard labels for the existing Q/A pairs used for training, the Yang et al. (2015) system employs UTS to retrieve the semantic types for each gold standard exact answer. A number of linguistic and semantic features are extracted from the tokens and concepts, including the lemma form of each token, the semantic type of each concept in the question, the dependency label of each token, combination of semantic type labels and dependency labels, etc., where the concepts are identified from MetaMap, LingPipe NER, and Apache OpenNLP Chunker[12] (noun phrases). A multi-class Logistic Regression classifier is trained using the LibLinear tool (Fan et al., 2008).

Candidate Answer Generation. Depending on the question type (general factoid/list question, `CHOICE` question, or `QUANTITY` question), the Yang et al. (2015) system applies different strategies to generate candidate answers. For general factoid/list questions, it generates a candidate answer using each concept identified by one of three

[4] https://github.com/oaqa/cse-framework/
[5] http://yaml.org/
[6] https://github.com/oaqa/baseqa/
[7] http://www.clearnlp.com
[8] http://alias-i.com/lingpipe/
[9] https://uts.nlm.nih.gov/home.html
[10] https://lucene.apache.org/
[11] http://www.csie.ntu.edu.tw/~cjlin/liblinear/
[12] https://opennlp.apache.org/

concept identification approaches. For CHOICE questions, it first identifies the "or" token in the question and its head token, which is most likely the *first option* in the list of candidate answers, and then finds all the children of the first option token in the parse tree that have a dependency relation of conj, which are considered to be *alternative options*. For QUANTITY questions, it identifies all the tokens that have a POS tag of CD in all relevant snippets.

Candidate Answer Scoring and Pruning. The Yang et al. (2015) system extends the approach used by Weissenborn et al. (Weissenborn et al., 2013) and defines 11 groups of features to capture how likely each candidate answer is the true answer for the question from different aspects, which includes answer type coercion, candidate answer occurrence count, name count, average overlapping token count, stopword count, overlapping concept count, token and concept proximity, etc. A Logistic Regression classifier is used to learn the scoring function, where the class is weighted by their frequencies. A simple threshold based pruning method is trained from the development dataset and applied to the list questions.

Besides incorporating a larger development data set, our OAQA system extends the Yang et al. (2015) system by integrating additional biomedical and general-purpose NLP annotators, and introducing trainable modules in more stages of the pipeline, such as using supervised methods in search result reranking, answer reranking, and yes/no answer prediction, which we will detail in the following sections. The architecture diagrams are illustrated in Figures 1, 2, and 3 in Appendix.

3 Concept Identification

We use the MetaMap and LingPipe concept identification modules with the GENIA model from Yang et al. (2015). However, due to the excessive noise introduced from the Apache OpenNLP Chunker based method, which extracts all noun phrases, we discard this approach. In addition, we integrate the TmTool biomedical concept identification RESTful service (Wei et al., 2016) for both the semantic type labeling of gold standard exact answers and question/snippet annotation, and also use C-Value (Frantzi et al., 2000), a frequent phrase mining method, to extract potential out-of-vocabulary multi-word terms.

3.1 TmTool for Annotating Questions, Snippets, and Answer Texts

The TmTool provides a standard Web service interface to annotate biomedical concepts using a number of state-of-the-art biomedical NLP parsers, which includes GNormPlus/SR4GN (for genes and species), tmChem (for chemicals), DNorm (for diseases), and tmVar (for mutations). Although it can only identify biomedical concepts belonging to any of these categories, they account for a great portion of the concepts used in the BioASQ corpus. In addition, many of these parsers utilize morphological features to estimate the likelihood of a term being a biomedical concept, rather than relying on an existing ontology like MetaMap, which makes it complementary to the existing tools in Yang et al. (2015).

TmTool supports three data exchange formats: PubTator (tab-separated), BioC (XML) and PubAnnotation (JSON). Since the PubTator format does not support DNorm annotation, and BioC format sometimes causes a single-sentence request to timeout (no response after 20 minutes) , we chose the robustest PubAnnotation format. We also found that the offsets returned from the TmTool RESTful service might not align well with original request texts, especially with tmChem trigger, and hence we implement an escape method to convert the texts into a TmTool compatible format by replacing some non-ASCII characters with their normalized forms, and removing special characters.

We use the TmTool to identify the biomedical concepts and annotate their semantic types from both the questions (in Phases A and B) and the relevant snippets (in Phase B) in the same manner as MetaMap. As the semantic type set of concepts has expanded to include TmTool concept types, the answer type prediction module should also be able to predict these additional semantic types. Therefore, we also use the TmTool to label the semantic types of the gold standard exact answers. In particular, we concatenate all the exact answers of each factoid or list question using commas, and send the concatenated string to the TmTool service, instead of each exact answer at a time. For example, if the gold standard exact answer is a list of strings: "NBEAL2", "GFI1B", "GATA1", then a single string "NBEAL2, GFI1B, GATA1" will be sent.

3.2 C-Value for Extracting Multi-Word Terms from Snippets

We treat the relevant snippets provided for each question in Phase B as a corpus, and we hypothesize that if a multi-word phrase is frequent in the corpus, then it is likely a meaningful concept. In order to extract not only high-frequency terms but also high-quality terms, a C-Value criterion (Frantzi et al., 2000) is introduced, which subtracts the frequency of its super terms from a term's own frequency. In this way, it returns the longer multi-word terms if two candidate terms overlap and have the same frequency. This approach only applies to a corpus, rather than a single sentence. Therefore, we only use this method to extract concepts from snippets. In the future, we may consider to collect a corpus relevant to the question, in order to apply the same idea to questions.

4 Collective Answer Reranking

We employ a collective answer reranking method aiming to boost the low-ranked candidate answers which share the same semantic type with high-ranked candidate answers for list questions, and use an adaptive threshold for pruning. The intuition is that list questions always ask for a list of concepts that have the same properties, which implies that the concepts usually have the same semantic types (e.g. all of them should be gene names). After the answer scoring step where a confidence score is assigned to each candidate answer *individually*, we can imagine the top candidate answers might have mixed types. For example, in a situation where the second answer is a disease, but the rest of the top-5 answers are all gene names, we should expect that the second answer should be down-ranked.

We use the same labels used for training the candidate answer scoring model, but incorporate features that measure how similar each answer is to the other top-ranked answers, which are detailed in Table 1. The *token distance* counts the number of intermediate tokens between the candidate answer tokens in the snippet text, and *Levenshtein edit distance* and *shape edit distance* measure the morphological similarities between the answer texts. *Common semantic type count* should "promote" the candidate answers that have a large number of semantic types in common with the top ranked answers.

For each candidate answer, we calculate a feature value, according to Table 1, for each other candidate answer in the input candidate list, and then we calculate the max/min/avg value corresponding to the top-k candidate answers. We use 1, 3, 5, 10 for k, and use Logistic Regression to train a binary classifier by down-sampling the negative instances to balance the training set. In addition to list questions, we also apply the method to factoid questions. In Section 7, we observe if the hypothesis also holds for factoid questions.

5 Learning to Answer Yes/No Questions

We consider the yes/no question answering problem as a binary classification problem, which allows to prioritize, weight, and blend multiple pieces of evidence from various approaches using a supervised framework. We list the sources of evidence (features) integrated into the system.

"Contradictory" concept. First, we hypothesize that if a statement is wrong, then the relevant snippets should contain some statements that are contradictory to the original statement, with some mentions of "contradictory" concepts or "antonyms". To identify pairs of contradictory concepts or antonyms is difficult given the resources that we have. Instead, we try to identify all the different concepts in the snippets that have the same semantic type as each concept in the original statement. For a given concept type, the more the unique concepts are found in both question and relevant snippets, or the less the concepts in the questions are found in the snippets, the more likely the original statement is wrong.

Formally, for a concept type t, we calculate a "contradictory" score as follows:

$$\frac{\sum_{s \in S} \sum_{c \in s} [\text{type}(c) = t]}{\sum_{c \in q} [\text{type}(c) = t] + \sum_{s \in S} \sum_{c \in s} [\text{type}(c) = t]}$$

where S is the set of snippets, q is the question, c is a concept mention, and $[\text{type}(c) = t]$ takes 1 if the concept c is type t and 0 otherwise. We derive the aggregated contradictory score from the concept type level scores using max/min/average statistics. We calculate a number of similar statistics to estimate how likely each snippet contradicts the original statement.

Overlapping token count. In case the concept identification modules fail to identify important concepts in either the original questions or relevant snippets, we also consider the difference of

No.	Feature
1	the original score from the answer scoring prediction
2	min/max/avg token distance between each pair of candidate answer occurrences
3	min/max/avg Levenshtein edit distance between each pair of candidate answer variant names
4	min/max/avg number (and percentage) of semantic types that each pair of candidate answers have in common
5	min/max/avg edit distance between each pair of candidate answer variant names after transformed into their *shape* forms (i.e. upper-case letters are replaced with 'A', lower-case letters are replaced with 'a', digits are replaced with '0', and all other characters are replaced with '-'.)

Table 1: Collective Answer Reranking Features

No.	Feature
1	"contradictory" concept count in the relevant snippets
2	overlapping token count in the relevant snippets
3	expected answer count in the relevant snippets
4	sentiment analysis via positive and negative word count of each relevant snippet
5	negation word count of each relevant snippet
6	question inversion

Table 2: Yes/No Question Answering Features

token mentions between the original question and the relevant snippets, instead of concepts.

Expected answer count. Not all concepts and tokens are equally important in the original questions. We find that many times the focus of a yes/no question is the last concept mention, which we denote as the *expected answer*. We count the frequency (and the percentage) that the expected answer is mentioned in the relevant snippets, as well as the frequency that concepts of the same type are mentioned.

Positive / negative / negation word count. Sometimes, an explicit sentiment is expressed in the relevant snippets to indicate how confident the author believes a statement is true or false. We use a simple dictionary [13] based method (Hu and Liu, 2004) for sentiment analysis, and we count whether and how many times each positive / negative word is mentioned in each snippet, then aggregate across the snippets using min / max / average. We also use a list of common English negation words[14] for negation detection, for simplicity. Intuitively, a high overlapping count with a high negative or negation count indicates that the original statement tends to be incorrect.

Question inversion. The question inversion method (Kanayama et al., 2012) answers a yes/no question by first converting it to a factoid question, then applies an existing factoid question answering pipeline to generate a list of alternate candidate answers, and finally evidence each candidate answer and rank them. If the expected answer in the original question is also ranked at the top among all candidates for the factoid question, then the statement is true.

In our system, we first assume the last concept mention corresponds to the expected answer. Therefore, its concept type(s) are also the answer type(s) of the factoid question, and all the synonyms of the concept are the answer variants. After the token(s) and concept mention(s) covered by the expected answer are removed from the original question and the question type is changed to `FACTOID`, we use the candidate answer generation and scoring pipeline for the factoid QA to generate and rank a list of candidate answers. Since annotating additional texts is computationally expensive, we do not retrieve any relevant snippets for the converted factoid questions, instead we only use the relevant snippets of the original yes/no questions (provided as part of the Phase B input). The rank and the score of the expected answer are used as question inversion features for yes/no question training.

We use a number of classifiers, e.g. Logistic Regression, Classification via Regression (Frank et

[13] `http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar`

[14] `http://www.enchantedlearning.com/wordlist/negativewords.shtml`

al., 1998), Simple Logistic (Landwehr et al., 2005) using LibLinear and Weka[15] tools (Hall et al., 2009), after we down-sampled the positive ("yes") instances. In Section 7, we report not only the performance of each method in terms of accuracy, but also accuracy on the "yes" questions and the "no" questions, since on an imbalanced dataset, a simple "all-yes" method is also a "strong" baseline.

6 Retrieval Result Reranking via Relevance Classification

For relevant document, concept, and snippet retrieval, we first retrieve a list of 100 candidate results, then we define a set of features to estimate the relevance of each candidate result and employ a standardized interface to incorporate these features to rerank the retrieval result, which is different from Yang et al. (2015), where each stage employs a different retrieval / reranking strategy.

First, we replace the GoPubMed services with local Lucene indexes as the response time is estimated to be at least 20 times faster, although the task performance could be slightly worse (.2762 in terms of MAP using the GoPubMed concept Web service vs. .2502 using the local Lucene index in our preliminary experiment for concept retrieval). The concept Lucene index was created by fusion of the same biomedical ontologies used by the GoPubMed services, where we create 3 text fields: concept name, synonyms, and definition, and 2 string fields: source (Gene Ontology, Disease Ontology, etc) and URI. The document Lucene index was created from the latest MEDLINE Baseline corpus[16] using Lucene's StandardAnalyzer. After a list of documents are retrieved and segmented into sections and sentences, the snippet Lucene index is then built in memory on-the-fly at the sentence level. The search query is constructed by concatenating all synonyms of identified concepts (enclosed in quotes) and all tokens that are neither covered by any concept mentions nor are stop words, where the most 5,000 common English words[17] are used as the stop list. Then, the query searches all text fields.

The standardized search result reranking interface allows each retrieval task to specify different scoring functions (features). The features that we used for concept, document, and snippet retrieval are listed in Table 3. For example, during concept search result reranking, we can check if each candidate concept is also identified in the question by a biomedical NER. During snippet reranking, we can also incorporate the meta information, such as section label (title, abstract, body text, etc.), offsets in the section, and length of each snippet. In the candidate retrieval step, we have used a query that combines all non stop words and concepts identified by all biomedical concept annotators, in order to guarantee high recall. However, it does not optimize the precision. For example, some annotators/synonym expansion services may falsely identify concepts and introduce noisy search terms, and some search fields tend to be less informative than others. Therefore, in the reranking step, we employ various query formulation strategies, e.g. only within certain text fields and/or only using a subset of concept annotators, and consider the search score and rank of each candidate search result as features.

For this year's evaluation, we use Logistic Regression to learn relevance classifiers for all the reranking tasks, after negative instances are down-sampled to balance the training set. In the future, we can also integrate learning-to-rank modules.

7 Results

Besides the proposed methods described in Sections 3 to 6, we also made a few minor changes to the Yang et al. (2015) system, including

1. separating the texts in the parentheses in all gold standard exact answers as synonyms before gold standard semantic type labeling and answer type prediction training,
2. introducing the "null" type for the exact answer texts if neither of the two concept search providers (TmTool or UTS) can identify,
3. and adding nominal features (e.g. answer type name, concept type name, etc.) in addition to existing numeric features (e.g. count, distance, ratio, etc.) for candidate answer scoring.

In this section, we first report the evaluation results using the held-out BioASQ 3B Batch 5 subset, and then we conduct a manual analysis using BioASQ 4B dataset for our yes/no question answering method.

We first compare the retrieval results (Phase A) In Table 4. We can see that the proposed retrieval

[15] http://www.cs.waikato.ac.nz/ml/weka/
[16] https://mbr.nlm.nih.gov/
[17] http://www.wordfrequency.info/

No.	Feature
	Concept
1	the original score from the concept retrieval step
2	overlapping concept count between the retrieval results and mentions annotated in the question
3	retrieval scores using various query formulation strategies
	Document
1	the original score from the document retrieval step
2	retrieval scores using various query formulation strategies
	Snippet
1	the score of the containing document
2	meta information, including the section label (abstract or title), binned begin/end offsets, binned length of the snippet, etc.
3	retrieval scores using various query formulation strategies

Table 3: Retrieval Result Reranking via Relevant Classification Features

Method	MAP	F1	Precision	Recall
		Concept		
LR	**.3216**	.0297	.0154	.5504
NO	.2361			
		Document		
LR	**.1364**	.0462	.0284	.2709
NO	.1003			
		Snippet		
NO	**.1073**	.0147	.0079	.3015
LR	.0826			

Table 4: Evaluation results on BioASQ 3B Batch 5 Phase A subset. LR represents a Logistic Regression based reranking method is used, and NO means no operation is performed, i.e. original retrieval scores are used.

result reranking method via Logistic Regression improves the performance of concept and document retrieval, but not snippet retrieval, which may be due to the fact that the input candidate snippets have been reranked using a similar set of features at the document reranking step, and no further information is provided during the subsequent snippet reranking step.

The exact answer generation results (Phase B) are shown in Table 5. We see that the best configuration for factoid question answering in terms of MRR is keeping the original feature set with no collective reranking. However, if additional features are used, then the collective reranking method can improve the performance, and achieve the highest lenient accuracy score.

To answer list questions, we tune the thresholds (hard threshold, or TP and ratio threshold, or RP) and report the results from the thresholds that maximize the F1 score. Although the best F1 score is achieved by incorporating additional features without collective reranking, and using a ratio-based pruning method, all other configurations without collective reranking have the lowest performance. In addition, we can see that additional features improve the performance in general, and after carefully tuning of the threshold and the ratio in the pruning step, we can achieve the same level of performance. We hypothesize that the proposed method (CR + RP) can renormalize the answer scores and is thus more robust than the baseline system (NO + TP) in the sense that the performance of the former approach is less sensitive to the predefined threshold, although the latter can sometimes outperform the former when the threshold is carefully tuned. We submitted two runs in BioASQ 4B Batch 5 evaluation: `oaqa-3b-5` and `oaqa-3b-5-e` for the proposed and baseline methods respectively (using the same thresholds), and initial evaluation result confirms our hypothesis.

Due to the imbalance between "yes" and "'no" questions, we report the mean negative and positive accuracy scores in addition to the overall accuracy for yes/no question answering. We can see the performance is very sensitive to the choice of the classifier. Using the same set of features, ClassificationViaRegression achieves the highest performance, with both negative and positive accuracy scores greatly above 0.5 (random). All other methods tend to predict "yes", which results in a high positive accuracy but a low (below 0.5) negative accuracy score.

Factoid			
Method	**Len.Ac.**	**MRR**	**Str.Ac.**
OF + NO	.5000	**.3843**	.3182
OF + CR	.4545	.3791	.3182
AF + CR	**.5455**	.3732	.2727
AF + NO	.5000	.3689	.2727
List			
Method	**F1**	**Precision**	**Recall**
AF + NO + RP	**.4291**	**.4449**	.4593
AF + CR + RP	.4246	.4045	**.4864**
AF + CR + TP	.3969	.4100	.4267
OF + CR + TP	.3704	.4231	.3645
OF + CR + RP	.3629	.3654	.3874
AF + NO + TP	.3463	.3840	.3677
OF + NO + RP	.3460	.3188	.4431
OF + NO + TP	.1461	.2639	.1183
Yes/No			
Method	**Ac.**	**Neg.Ac.**	**Pos.Ac.**
CVR	**.7143**	**.7778**	.6842
SL	**.7143**	.4444	**.8421**
AY	.6786	.0000	1.0000
LR	.5357	.2222	.6842

Table 5: Evaluation results on BioASQ 3B Batch 5 Phase B subset. OF and AF represent Original or Additional features are used in training and predicting answer scorers for factoid and list questions. CR represents the Logistic Regression based Collective Reranking is used. TP means a hard Threshold is used to prune the answer, whereas RP uses the relative Ratio to the maximum score. CVR and SL are ClassificationViaRegression and SimpleLogistic classifiers from the Weka toolkit. AY means "All-Yes", a simple but strong baseline, in terms of accuracy.

8 Analysis

From Section 7, we see that, despite integration of various sources of evidence, the current yes/no question answering system is still unreliable. We conducted a manual analysis of our yes/no question answering method using BioASQ 4B dataset based on our own judgment of yes or no, which may not be consistent with the gold standard.

We found the BioASQ 4B dataset is more imbalanced than the development dataset, where we only identified six questions from all five test batches that have a "no" answer. We applied the proposed yes/no QA method to the six questions. Among these questions, three are correctly predicted (namely, "Is macitentan an ET agonist?", "Does MVIIA and MVIIC bind to the same calcium channel?", and "Is the abnormal dosage of ultraconserved elements disfavored in cancer cells?"), and the answers to the other three questions are wrong. We conduct an error analysis for the false positive predictions.

The first false positive question is "Are adenylyl cyclases always transmembrane proteins?" The key to this question is the recognition of the contradictory concept pair "transmembrane" and "soluble" or "transmembrane adenylyl cyclase (tmAC)" and "soluble AC". This requires first correctly identifying both terms as biomedical concepts and then assigning correct semantic type labels to them, where the latter can only be achieved using MetaMap and TmTool. MetaMap correctly identified "transmembrane proteins" in the question and assigned a semantic label of "Amino Acid, Peptide, or Protein", and identified "soluble adenylyl cyclase" in the snippet and assigned a semantic label of "Gene or Genome". Due to the mismatch of semantic types "Amino Acid, Peptide, or Protein" and "Gene or Genome", the system fails to recognize the contradiction.

In fact, we found that the same problem also happened during the answer type prediction and answer scoring steps, e.g. the question may be predicted to ask for a "Gene or Genome", but the candidate answer is often labeled as "Amino Acid, Peptide, or Protein" by MetaMap/UTS. Because of the interchangeable use of "Amino Acid, Peptide, or Protein" and "Gene or Genome" terms, we might consider to treat them as one type. Moreover, the universal quantifier "always" also plays an important role, in contrast to a question with an existential quantifier such as "sometimes", which the current system has not captured yet. However, this is not the main reason of the failure, since we assume the relevant snippets will rarely mention "soluble AC" if the question asks for whether "transmembrane" exists.

The second false positive question is "Can chronological age be predicted by measuring telomere length?" This should be an easy one, because we can find a negation cue "cannot" in the snippet "telomere length measurement by real-time quantitative PCR cannot be used to predict age of a person". The system integrates two types of negation cue related features: the negation cue count and the existence of a particular negation cue. We found the system correctly identified

and counted the negation cue. Therefore, we suspect the classifier did not optimize the combination of features. Furthermore, we need to observe whether our hypothesis that the gold standard answer (yes or no) is strongly correlated with the negation word occurrence in the relevant snippets is true using the development set.

The third false positive question is "Does the 3D structure of the genome remain stable during cell differentiation?" The key to this question is the word "stable", which requires biomedical, esp. genomics, knowledge to understand what "stable" means in the context of genome structure. The word "stable" is mentioned in one of the snippets "the domains are stable across different cell types", which however does not answer the question. Useful contradictory keywords that we find in the relevant snippets include "reorganization", "alteration", "remodelling", etc. MetaMap/UTS identified "stable" as a concept of semantic type "Qualitative Concept", whereas it labeled "reorganization" as a "Idea or Concept" and missed "alternation" and "remodelling". It suggests that our contradictory concept based method works the best if the focus is factoid (entities), but the current knowledge base can hardly support identification of contradictory properties or behaviors.

We focus on 4B Batch 5 subset for error analysis of false negative examples. In fact, the cases for false negative questions are more diverse, which makes it more difficult to find the causes of failures. One reason is that some snippets contain multiple sentences or clauses, and only one is crucial to answer the question, while others can negatively influence the results. For example, the snippet "OATP1B1 and OATP1B3-mediated transport of bilirubin was confirmed and inhibition was determined for atazanavir, rifampicin, indinavir, amprenavir, cyclosporine, rifamycin SV and saquinavir." has two clauses, but the second one ("and inhibition..."), although is not relevant to the question, introduces other chemical names that confuse the classifier. Another problem is lack of understanding of specificity and generality between concepts, e.g. "encephalopathy" in the question is considered a different concept from "Wernicke encephalopathy" mentioned in the snippets, both belonging to the same disease category. The classifier believed another disease name is mentioned to contradict the statement.

We found that yes/no questions are more difficult to answer than factoid and list questions, since there can be many different ways to support or oppose a statement. Although the problem can be simply viewed as a binary classification problem, due to the fact that a limited number of relevant snippets are provided, simple token or phrase level retrieval and statistics can hardly solve the problem. Instead, we believe that reliably answering yes/no questions requires deeper linguistic and semantic understanding of the questions and relevant snippets, which includes leveraging semantic networks of concepts to identify antonyms, hypernyms, and hyponyms, and utilizing dependency relations between the concepts, as well as sentiment analysis of the facts.

9 Conclusion

This paper describes the OAQA system evaluated in the BioASQ 4B Question Answering track. We first present the overall architecture of the system, and then focus on describing the main differences from the Yang et al. (2015) system, including two concept identification modules: TmTool and C-value based multi-word term extractor, collective answer reranking, yes/no question answering approach, and a standardized retrieval result reranking method via relevant classification. We report our initial evaluation results on 3B Batch 5 subset show the effectiveness of the proposed new methods, and since the yes/no question answering approach is unsatisfactory, we further conduct an error analysis for yes/no questions using 4B subset.

As we mention in earlier sections, to further improve the retrieval performance, we may use learning-to-rank methods to rerank the retrieval results. For exact answer generation, esp. for yes/no questions, we believe a deeper linguistic and semantic analysis of both questions and relevant snippets are necessary. Our preliminary experiment suggested that the word2vec (Mikolov et al., 2013) based method did worse than the KB based method in modeling the semantics of entities. We plan to study whether the former is complementary to the latter in representing the semantics of biomedical properties and event mentions.

Acknowledgments

We thank Ying Li, Xing Yang, Venus So, James Cai and the other team members at Roche Innovation Center New York for their support of OAQA and biomedical question answering research and development.

References

Alan R Aronson. 2001. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In *Proceedings of the AMIA Symposium*, page 17. American Medical Informatics Association.

Jinho D Choi and Martha Palmer. 2011. Getting the most out of transition-based dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 687–692. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9(Aug):1871–1874.

David Ferrucci, Eric Nyberg, James Allan, Ken Barker, Eric Brown, Jennifer Chu-Carroll, Arthur Ciccolo, Pablo Duboue, James Fan, David Gondek, et al. 2009. Towards the open advancement of question answering systems. Technical Report RC24789 (W0904-093), IBM Research Division.

Eibe Frank, Yong Wang, Stuart Inglis, Geoffrey Holmes, and Ian H Witten. 1998. Using model trees for classification. *Machine Learning*, 32(1):63–76.

Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. 2000. Automatic recognition of multi-word terms:. the c-value/nc-value method. *International Journal on Digital Libraries*, 3(2):115–130.

Elmer Garduno, Zi Yang, Avner Maiberg, Collin McCormack, Yan Fang, and Eric Nyberg. 2013. Cse framework: A uima-based distributed system for configuration space exploration. In *UIMA@GSCL'2013*, pages 14–17.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Hiroshi Kanayama, Yusuke Miyao, and John Prager. 2012. Answering yes/no questions via question inversion. In *COLING*, pages 1377–1392. Citeseer.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Junichi Tsujii. 2003. Genia corpusa semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl 1):i180–i182.

Niels Landwehr, Mark Hall, and Eibe Frank. 2005. Logistic model trees. *Machine Learning*, 59(1-2):161–205.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. 2015. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16(1):138.

Karin Verspoor, Kevin Bretonnel Cohen, Arrick Lanfranchi, Colin Warner, Helen L Johnson, Christophe Roeder, Jinho D Choi, Christopher Funk, Yuriy Malenkiy, Miriam Eckert, et al. 2012. A corpus of full-text journal articles is a robust evaluation tool for revealing differences in performance of biomedical natural language processing tools. *BMC bioinformatics*, 13(1):1.

Chih-Hsuan Wei, Robert Leaman, and Zhiyong Lu. 2016. Beyond accuracy: creating interoperable and scalable text-mining web services. *Bioinformatics*, pages 1–4.

Dirk Weissenborn, George Tsatsaronis, and Michael Schroeder. 2013. Answering factoid questions in the biomedical domain. In *BioASQ'2013*.

Zi Yang, Elmer Garduno, Yan Fang, Avner Maiberg, Collin McCormack, and Eric Nyberg. 2013. Building optimal information systems automatically: Configuration space exploration for biomedical information systems. In *CIKM'2013*, pages 1421–1430.

Zi Yang, Niloy Gupta, Xiangyu Sun, Di Xu, Chi Zhang, and Eric Nyberg. 2015. Learning to answer biomedical factoid & list questions: Oaqa at bioasq 3b. In *CLEF'2015 (Working Note)*.

Appendix

Listing 1: ECD main descriptor for factoid and list QA in batch 5 in Phase B

```
# execute
#      mvn exec:exec -Dconfig=bioasq.
    phase-b-test-factoid-list
# to test the pipeline

configuration:
  name: phase-b-test-factoid-list
  author: ziy

persistence-provider:
  inherit: baseqa.persistence.
    local-sqlite-persistence-provider

collection-reader:
  inherit: baseqa.collection.json.
    json-collection-reader
  dataset: BIOASQ-QA
  file:
    - input/4b-5-b.json
  type: [factoid, list]
  persistence-provider: |
    inherit: baseqa.persistence.
     local-sqlite-persistence-provider

pipeline:
  - inherit: ecd.phase
    name: question-parse
    options: |
      - inherit: bioqa.question.parse.
     clearnlp-bioinformatics

  - inherit: ecd.phase
    name: question-concept-metamap
    options: |
      - inherit: bioqa.question.concept.
     metamap-cached

  - inherit: ecd.phase
    name: question-concept-tmtool
    options: |
      - inherit: bioqa.question.concept.
     tmtool-cached

  - inherit: ecd.phase
    name: question-concept-lingpipe-genia
    options: |
      - inherit: bioqa.question.concept.
     lingpipe-genia

  - inherit: ecd.phase
    name: question-focus
    options: |
      - inherit: baseqa.question.focus

  - inherit: ecd.phase
    name: passage-to-view
    options: |
      - inherit: baseqa.evidence.passage-to-view

  - inherit: ecd.phase
    name: evidence-parse
    options: |
      - inherit: bioqa.evidence.parse.
     clearnlp-bioinformatics

  - inherit: ecd.phase
    name: evidence-concept-metamap
    options: |
      - inherit: bioqa.evidence.concept.
     metamap-cached

  - inherit: ecd.phase
    name: evidence-concept-tmtool
    options: |
      - inherit: bioqa.evidence.concept.
     tmtool-cached

  - inherit: ecd.phase
    name: evidence-concept-lingpipe-genia
    options: |
      - inherit: bioqa.evidence.concept.
     lingpipe-genia

  - inherit: ecd.phase
    name: evidence-concept-frequent-phrase
    options: |
      - inherit: baseqa.evidence.concept.
     frequent-phrase

  - inherit: ecd.phase
    name: concept-search-uts
    options: |
      - inherit: bioqa.evidence.concept.
     search-uts-cached

  - inherit: ecd.phase
    name: concept-merge
    options: |
      - inherit: baseqa.evidence.concept.merge

  - inherit: ecd.phase
    name: answer-type
    options: |
      - inherit: bioqa.answer_type.
     predict-liblinear-null

  - inherit: ecd.phase
    name: answer-generate
    options: |
      - inherit: bioqa.answer.generate.generate

  - inherit: ecd.phase
    name: answer-modify
    options: |
      - inherit: baseqa.answer.modify.modify

  - inherit: ecd.phase
    name: answer-score
    options: |
      - inherit: bioqa.answer.score.
     predict-liblinear

  - inherit: ecd.phase
    name: answer-collective-score
    options: |
      - inherit: bioqa.answer.collective_score.
     predict-liblinear
      - inherit: base.noop

  - inherit: ecd.phase
    name: answer-prune
    options: |
      - inherit: baseqa.answer.modify.pruner

#  - inherit: baseqa.cas-serialize

post-process:
  # submission
  - inherit: bioasq.collection.json.
     json-cas-consumer
```

Listing 2: ECD main descriptor for yes/no QA in batch 5 in Phase B

```
# execute
#      mvn exec:exec -Dconfig=bioasq.
    phase-b-test-yesno
# to test the pipeline

configuration:
  name: phase-b-test-yesno
  author: ziy

persistence-provider:
  inherit: baseqa.persistence.
    local-sqlite-persistence-provider

collection-reader:
  inherit: baseqa.collection.json.
    json-collection-reader
  dataset: BIOASQ-QA
  file:
    - input/4b-5-b.json
  type: [yesno]
  persistence-provider: |
```

```
    inherit: baseqa.persistence.
     local-sqlite-persistence-provider

pipeline:
  - inherit: ecd.phase
    name: question-parse
    options: |
      - inherit: bioqa.question.parse.
     clearnlp-bioinformatics

  - inherit: ecd.phase
    name: question-concept-metamap
    options: |
      - inherit: bioqa.question.concept.
     metamap-cached

  - inherit: ecd.phase
    name: question-concept-tmtool
    options: |
      - inherit: bioqa.question.concept.
     tmtool-cached

  - inherit: ecd.phase
    name: question-concept-lingpipe-genia
    options: |
      - inherit: bioqa.question.concept.
     lingpipe-genia

  - inherit: ecd.phase
    name: passage-to-view
    options: |
      - inherit: baseqa.evidence.passage-to-view

  - inherit: ecd.phase
    name: evidence-parse
    options: |
      - inherit: bioqa.evidence.parse.
     clearnlp-bioinformatics

  - inherit: ecd.phase
    name: evidence-concept-metamap
    options: |
      - inherit: bioqa.evidence.concept.
     metamap-cached

  - inherit: ecd.phase
    name: evidence-concept-tmtool
    options: |
      - inherit: bioqa.evidence.concept.
     tmtool-cached

  - inherit: ecd.phase
    name: evidence-concept-lingpipe-genia
    options: |
      - inherit: bioqa.evidence.concept.
     lingpipe-genia

  - inherit: ecd.phase
    name: evidence-concept-frequent-phrase
    options: |
      - inherit: baseqa.evidence.concept.
     frequent-phrase

  - inherit: ecd.phase
    name: concept-search-uts
    options: |
      - inherit: bioqa.evidence.concept.
     search-uts-cached

  - inherit: ecd.phase
    name: concept-merge
    options: |
      - inherit: baseqa.evidence.concept.merge

  - inherit: ecd.phase
    name: answer-yesno
    options: |
      - inherit: bioqa.answer.yesno.
     predict-weka-other
     - inherit: baseqa.answer.yesno.all-yes

post-process:
  # submission
  - inherit: bioasq.collection.json.
     json-cas-consumer
```

Listing 3: ECD main descriptor for retrieval in batch 5 in Phase A

```
# execute
#      mvn exec:exec -Dconfig=bioasq.phase-a-test
# to test the pipeline

configuration:
  name: phase-a-test
  author: ziy

persistence-provider:
  inherit: baseqa.persistence.
     local-sqlite-persistence-provider

collection-reader:
  inherit: baseqa.collection.json.
     json-collection-reader
  dataset: BIOASQ-QA
  file:
    - input/4b-5-a.json
  persistence-provider: |
    inherit: baseqa.persistence.
     local-sqlite-persistence-provider

pipeline:
  - inherit: ecd.phase
    name: question-parse
    options: |
      - inherit: bioqa.question.parse.
     clearnlp-bioinformatics

  - inherit: ecd.phase
    name: question-concept-metamap
    options: |
      - inherit: bioqa.question.concept.
     metamap-cached

  - inherit: ecd.phase
    name: question-concept-tmtool
    options: |
      - inherit: bioqa.question.concept.
     tmtool-cached

  - inherit: ecd.phase
    name: question-concept-lingpipe-genia
    options: |
      - inherit: bioqa.question.concept.
     lingpipe-genia

  - inherit: ecd.phase
    name: concept-search-uts
    options: |
      - inherit: bioqa.evidence.concept.
     search-uts-cached

  - inherit: ecd.phase
    name: concept-merge
    options: |
      - inherit: baseqa.evidence.concept.merge

  - inherit: ecd.phase
    name: abstract-query-primary
    options: |
      - inherit: baseqa.abstract_query.token-concept

  # concept
  - inherit: ecd.phase
    name: concept-retrieval
    options: |
      - inherit: bioqa.concept.retrieval.
     lucene-bioconcept

  - inherit: ecd.phase
    name: concept-rerank
    options: |
      - inherit: bioqa.concept.rerank.
     predict-liblinear

  # document
  - inherit: ecd.phase
    name: document-retrieval
    options: |
      - inherit: bioqa.document.retrieval.
     lucene-medline

  - inherit: ecd.phase
```

```
    name: document-rerank
    options: |
      - inherit: bioqa.document.rerank.
      predict-liblinear

  # snippet
  - inherit: ecd.phase
    name: passage-retrieval
    options: |
      - inherit: bioasq.passage.retrieval.
      document-to-passage

  - inherit: ecd.phase
    name: passage-rerank
    options: |
      - inherit: bioqa.passage.rerank.
      predict-liblinear
      - inherit: base.noop

post-process:
  # submission
  - inherit: bioasq.collection.json.
      json-cas-consumer
```

Listing 4: ECD component descriptor of bioqa.answer.collective_score.liblinear-predict

```
inherit: baseqa.learning_base.classifier-predict

candidate-provider: 'inherit: baseqa.answer.score.
    candidate-provider'
scorers: |
  - inherit: baseqa.answer.collective_score.scorers.
    original
  - inherit: baseqa.answer.collective_score.scorers.
    distance
  - inherit: baseqa.answer.collective_score.scorers.
    edit-distance
  - inherit: baseqa.answer.collective_score.scorers.
    type-coercion
  - inherit: baseqa.answer.collective_score.scorers.
    shape-distance
classifier: 'inherit: bioqa.answer.collective_score.
    liblinear-classifier'
feature-file: result/
    answer-collective-score-predict-liblinear.tsv
```

Listing 5: ECD component descriptor of bioqa.answer.yesno.predict

```
inherit: baseqa.answer.yesno.predict

scorers: |
  - inherit: baseqa.answer.yesno.scorers.
    concept-overlap
  - inherit: bioqa.answer.yesno.scorers.
    token-overlap
  - inherit: baseqa.answer.yesno.scorers.
    expected-answer-overlap
  - inherit: baseqa.answer.yesno.scorers.sentiment
  - inherit: baseqa.answer.yesno.scorers.negation
  - inherit: bioqa.answer.yesno.scorers.
    question-inversion
```

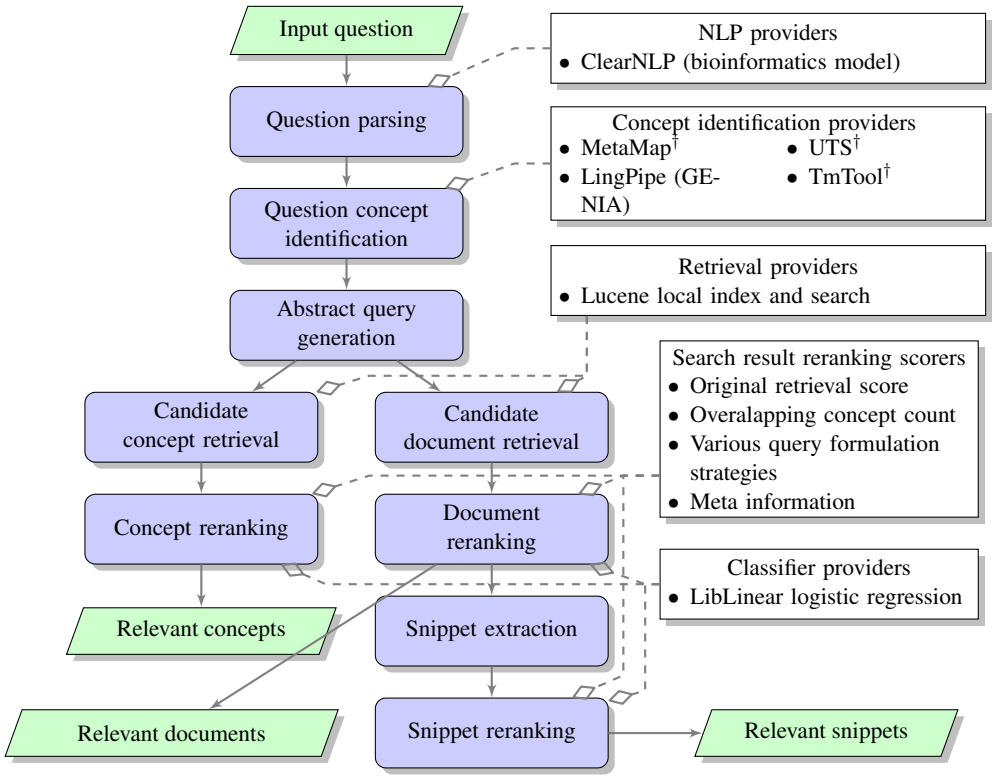

Figure 1: Retrieval (Phase A) pipeline diagram. † represents a provider that requires accessing external Web services.

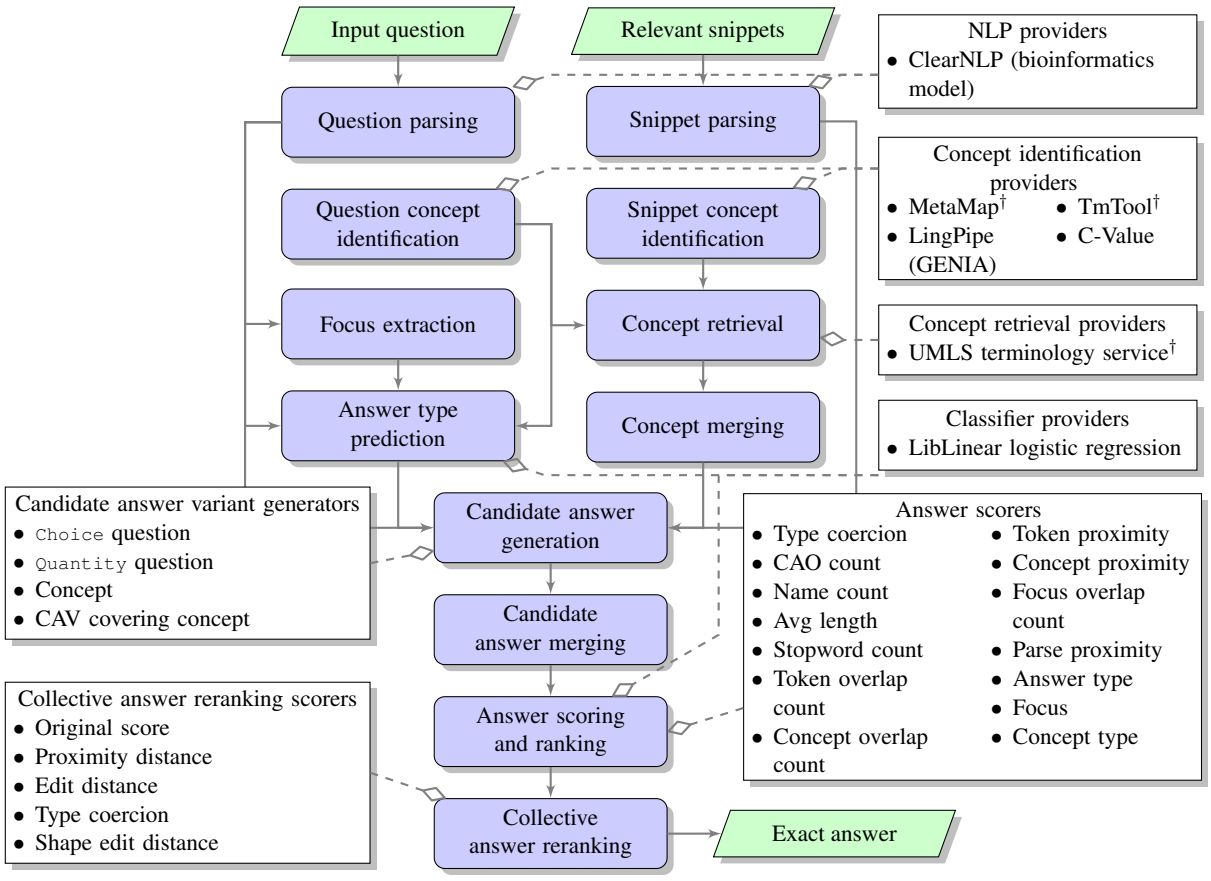

Figure 2: Factoid and list question answering (Phase B) pipeline diagram. † represents a provider that requires accessing external Web services.

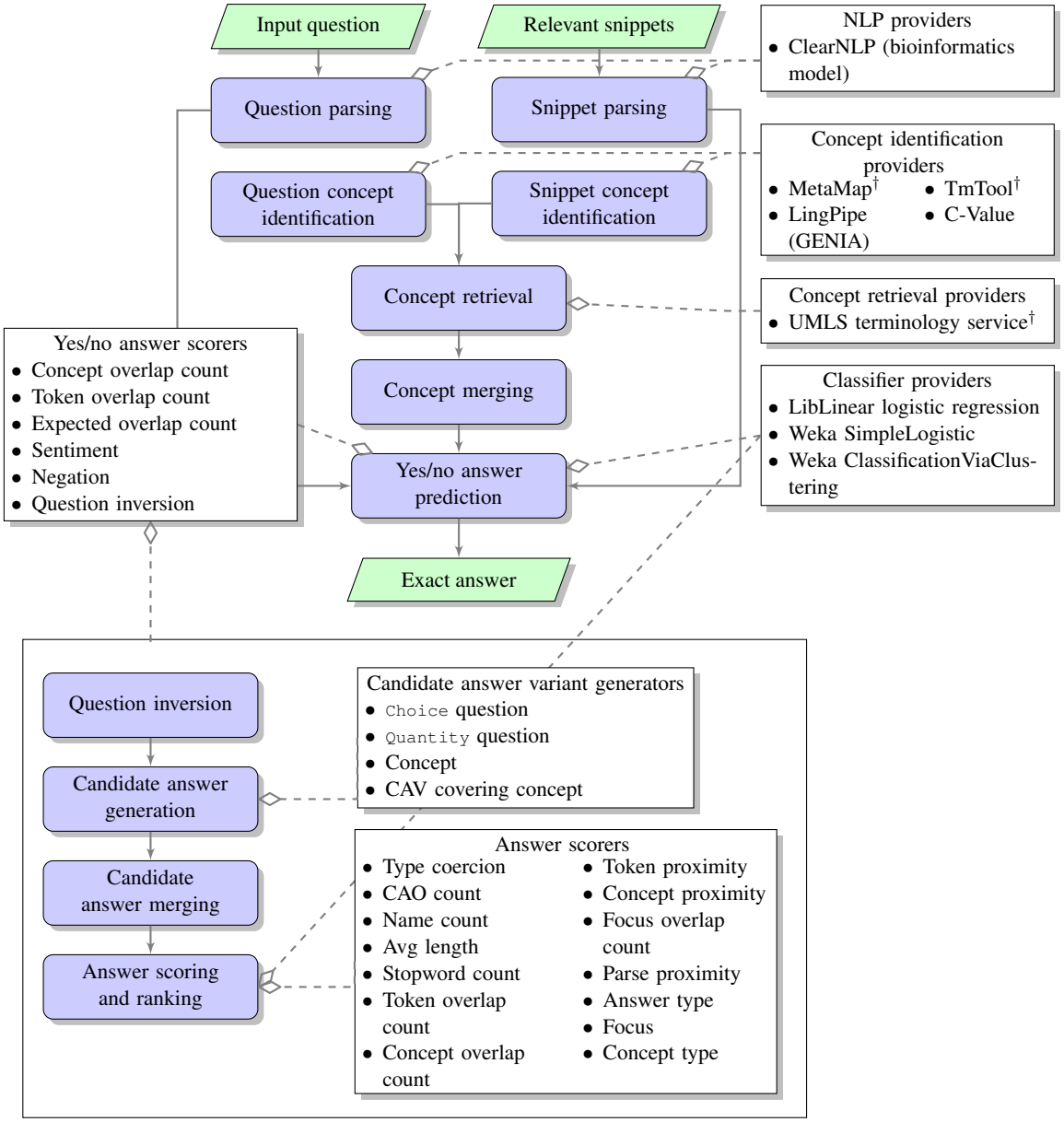

Figure 3: Yes/no question answering (Phase B) pipeline diagram. † represents a provider that requires accessing external Web services.