Meta-Knowledge Annotation for Efficient Natural-Language Question-Answering

TonyVeale

Computer Science Dept., University College Dublin
Tony.veale@ucd.ie

Abstract: A recent trend in the exploitation of unstructured text content is the use of natural language question answering (NLQA) systems. NLQA is an elaboration of traditional information retrieval techniques for satisfying a user's information needs, where the goal is not simply to retrieve relevant documents but to additionally extract specific passages and semantic entities from these documents as candidate answers to a natural language question. NLQA is thus a tight integration of natural language processing (NLP), information retrieval (IR) and information extraction (IE) designed to circumvent the deep and brittle analysis of questions in favor of shallow but robust comprehension, to ultimately achieve a broad domain question-answering competence. It is argued here that the key to achieving good quality answers in a high-throughput setting lies in a system's ability to construct rich queries that incorporate knowledge from multiple sources.

1 Question Answering Systems

Question-answering systems that exploit information-retrieval are predicated on two major assumptions:

1. Most interesting questions have already been answered somewhere, so NLQA can be conceived as a shallow textual search for existing answers rather than a deep logical proof of novel ones (see [1], [5]).
2. The textual form of a question tends to resemble the textual form of its answer. Thus an IR query constructed from a question text will be effective in retrieving documents that contain answers.

Though neither of these assumptions is defensible on theoretical grounds, there are strong empirical grounds for adopting them both, insofar as they yield computationally tractable systems that offer satisfactory service to users. These assumptions shape the general approach to IR-based NLQA as follows:

1. Given a user question *Q*, generate a natural-language representation *nlp(Q)*
2. From *nlp(Q)*, generate an information-retrieval query *ir(nlp(Q))*

3. Use *ir(nlp(Q))* to retrieve *D* documents from an authoritative text archive
4. If *D* is too small or too large, modify *ir(nlp(Q))* accordingly and return to 3
5. Analyze the contents of *D* to derive *A* candidate answer passages from *D*
6. Perform semantic labeling to extract the set of named-entities *E* in *A*
7. If *E* is too small or not compatible with *nlp(Q)*, modify *ir(nlp(Q))* accordingly and return to 3
8. Rank and display <*A*, *E*> according to the semantic target of *Q* (i.e., who, when, where, etc.)

NLQA systems primarily differ as to the complexity of *ir(nlp(Q))* and the ideal sizes for *D* and *A*, the number of retrieved documents and candidate answer passages. When *ir(nlp(Q))* is richly structured and highly constrained, *D* is typically small and the work load of the IE component is low. But when *ir(nlp(Q))* is simple (a bag of conjoined words, say), then *D* is typically large and the IE component can become a bottleneck, particularly in a commercial setting where question throughput is almost as important as answer precision. To reduce recall to a manageable level, a system can make *ir(nlp(Q))* as constrained as possible. However, this can have the opposite effect of staunching recall completely, so that the system is unable to provide any answers at all. To strike an effective balance, an NLQA system employs an iterative search to incrementally modify *ir(nlp(Q))* until sufficient documents can be found to supply a convincing answer. NLQA is thus a heuristic search process through the space of possible *ir(nlp(Q))* formulations, until one is found to semantically fit the available text data. The unstated goal of commercial NLQA is to guide this tour of the search-space in as efficient a manner as possible without sacrificing competence.

The formulation of *nlp(Q)* has been discussed in depth elsewhere [3]; it suffices to state here that it presupposes a shallow NLP analysis in which multiword phrase boundaries are recognized, parts-of-speech are assigned to each word or phrase, head-words are identified and named entities (such as people and companies) are appropriately tagged. Instead, this paper focuses on a means of formulating *ir(nlp(Q))* that provides as much knowledge as possible to the heuristic processes that will maximize NLQA throughput, by intelligently directing the modification and exploitation of *ir(nlp(Q))*.

2 Annotations and Strategies

The iterative nature of NLQA suggests that it is a non-deterministic integration of NL, IR and IE: when IE is unsatisfactory, IR must be regenerated which involves a reconsideration of NL. This non-determinism requires that each component maintain an internal state so that it may be reactivated as the search process re-iterates. This not only increases and complicates the mutual dependence between components, it introduces considerable overhead from a session-tracking perspective into a high-volume NLQA system. To make these components mutually independent and largely state-free, *ir(nlp(Q))* is thus conceived as a one-off meta-query, whose annotations contain knowledge to drive future modification and evaluation processes without the involvement of the NL and IR components. Since these annotations mark the relative importance of each part of the query (as determined by NL analysis), a richer scoring mechanism can be used to evaluate retrieved answers. This approach has broad implications, since it transforms *ir(nlp(Q))* from a simple by-product of knowledge to an active knowledge component in its own right.

Annotations primarily allow a query to be iteratively modified in two ways:

1. A non-focal term can be *pruned* if it is does more to reduce recall than increase precision.
2. A term can be *expanded* by adding synonyms and other correlated disjuncts.

The purpose of both strategies is to increase the recall of a tightly constrained query. We do not consider strategies that also reduce recall, since this leads to a combinatorial search space of possible modification sequences. Thus we assume that *ir(nlp(Q))* as initially formulated is maximally constrained, so that it contains each non-stopword of the question, it uses proximity operators to tie modifier terms to their heads, and uses phrasal operators to preserve the internal ordering of named entities.

As for meta-knowledge, the pruning strategy requires that a query be annotated as to the answer utility of each keyword or query fragment. Utility is discussed in the next section, so for now it suffices to note that certain elements of a question are more focal to the user's information need than others, and that the most focal elements should be the last to be pruned from a query. The expansion strategy requires that all possible disjunctions (synonyms, holonyms, partonyms, etc.) are already present in the query as annotations, so that expansion is an in-place conversion of an annotation to a query operator. We do not consider here strategies that weaken the proximity constraints in distance-based operators, assuming instead for simplicity that these constraints have sufficient breathing space in their original specification.

The query language used is that of Inquery, which uses a prefix notation not unlike that of Prolog [2]. The conjunction operator is *#and*, the disjunction operator is *#or*, the phrase or contiguity operator is $\#od_n$ (where n >= 1 is the allowable gap between words), and the proximity or window operator is $\#uw_n$ (where n is the maximum window size). Additionally, the *#syn* operator is used to indicate synonymy at the word and phrase level. To this basic set we add following meta-operators, which can be used to insert annotations at any level of a query:

1. *#meta_alpha*	Marks a keyword or fragment as having high answer utility
2. *#meta_beta*	Marks a keyword or fragment as having indifferent utility
3. *#meta_gamma*	Marks a keyword or fragment as having low answer utility
4. *#meta_syn*	Specifies possible disjunctions for the first argument term
5. *#meta_utility*	Associates a numeric utility with a term or fragment
6. *#meta_target*	Specifies a semantic type that must be present in an answer

When a meta-query is used to retrieve documents, the annotations are recursively removed by discarding all but the first argument of every meta-operator. Since meta-operators are not idempotent, a term annotated by two nested *#meta_alpha* operators is attributed a much higher answer-utility than a term annotated with just one. Likewise, a term annotated with two *#meta_gamma* operators possesses little utility.

The expansion strategy works by simply converting a *#meta_syn* operator into a *#syn* operator. As a result, each annotation argument is transformed from meta-data into query-data, by becoming a *#syn* argument that is actually used for document retrieval. Therefore, expansion only applies to *#meta_syn* nodes that have literal query fragments or terms as arguments. Pruning is also straightforward to perform, and is not as drastic a strategy as it might first seem. This is because a question essentially contains two kinds of content words: those that are best used to retrieve candidate answers, and those that are best used to evaluate the semantic fit of those answers to the question. While these kinds are not mutually exclusive, some kinds of words are clearly suited to one purpose rather than another. For instance, nouns are generally best suited for retrieval since they often determine the domain of discourse, while adverbs are best suited to evaluation since they tend to reflect a subjective opinion about a situation. So when a term is pruned from a query, it is not actually discarded but merely moved, from the retrieval set to the evaluation set. Indeed, since evaluation occurs post-retrieval, greater computational effort can at this point be spent fitting an evaluation term to an answer (e.g., using WordNet-based similarity measures [4]).

3 Answer Utility

Pruning is thus used to ensure that those terms that most reduce recall while least improving precision are placed in a question's evaluation set rather than its retrieval set, despite their linguistic class. For example, some terms are used to facilitate the framing of the question rather than to shape its semantic content, as in the use of the word "think" in "What does Darwin think about acquired characteristics". But to recognize these framing terms for what they are – contextual noise – requires a powerful grammar and rich model of language use. Alternately then, we do not exclude them from the query but contrive to ensure that such terms are amongst the first to be pruned. This contrivance takes the form of *answer utility*, a measure that attempts to quantify the harm that would be done to answer quality if a retrieval term were to be pruned.

Several factors combine to yield the answer utility of a term. Syntactic utility is an estimate of utility based on the syntactic function of a term, while semantic utility estimates the contribution of a term to the meaning of the question and thus, indirectly, to the meaning of the answer. Named entities such as people names, companies and products have the highest syntactic utility, since they almost always establish the focus of the user's information need. So for example, in the question "Who is the CEO of Vantive", the term "Vantive" has higher syntactic utility than "CEO", which is to say that we should prefer to prune the latter before the former. We thus consider named entities to be *double-alpha* terms, which means that these terms are annotated with two nested *#meta_alpha* operators in a query. Of slightly less utility are nouns, which are the alpha-terms of a question since they serve to anchor it in a particular domain, while verbs and adjectives are viewed indifferently as beta-terms. Finally, adverbs as treated the gamma-terms, since the meaning carried by an adverb in a question may be expressed in an answer via an adjective, a verb or a noun; this mutability makes for poor retrieval so this is reflected in a lower perceived utility.

Semantic utility can be estimated when an NLQA system has direct access to the indices of the underlying IR engine, so that term co-occurrence probabilities and estimated mutual information measures can be used to determine the discrimination power of a term with respect to the question as a whole. Though space precludes a full discussion here, the general method is briefly outlined. First, it is necessary to capture the semantic focus of the question. Using conditional term probabilities, the focus can be modeled as the term that best predicts the question as a whole. (e.g., in “Who was the captain of the Titanic”, the focus is Titanic, not captain). Next, the focality of every other term in the question is estimated, by calculating the mutual information between each term and the focus. The redundancy of each term is then estimated, again using conditional term probabilities, as the extent to which the question as a whole predicts the given term (redundancy is thus the converse of focality). Finally, the semantic utility of each term is estimated as a function of focality and redundancy; there are many ways this function can be formulated, but in general, utility will be proportional to focality and inversely proportional to redundancy.

The operator *#meta_utility* is used to annotate the leaf terms of a query with its corresponding measure of semantic utility. Higher-level annotations will additionally capture the syntactic utility of these terms, so that a recursive measure of utility can be recursively calculated for any fragment of the query. This allows the fragment of weakest utility to be determined at any point and pruned (but note: the leaf terms of the pruned fragment are placed in the evaluation set for any retrieved answers).

4 Cost-Driven Search

NLQA is a hill-climbing search through the space of possible text answers to a question. The start state of this search is determined by the query *ir(nlp(Q))*, which is initially constructed to be maximally constrained and gradually weakened via expansion and pruning. Though NLQA is fundamentally an IR process, it wields a key advantage over conventional IR, since a natural language question specifies a semantic target (e.g. PERSON for “who” questions, PLACE for “where” questions) that serves as valuable meta-knowledge for the search. It allows a system to determine whether the goal state has truly been found, or whether more query modification is required even when apparently relevant documents have already been found. Thus, if an entity of the desired type cannot be extracted from an otherwise relevant document set, NLQAs system will prolong a search that an IR process would instead terminate.

Pruning and expansion are the state-traversal operators that allow NLQA to navigate the space of possible answers. Each strategy is assumed to select and modify a single node of the query at a time, but because pruning is destructive, the order of modifications needs to be explicitly scheduled. A governing principle of *least cost modification* is needed to determine the order of this schedule, where this cost reflects the loss of answer utility inflicted on the query by a modification. For pruning, the modification cost is equivalent to the answer utility of the term to be pruned, so the least useful terms are always pruned first. For expansion, the cost is an estimate of the decline in term utility when less reliable disjuncts are introduced (as discussed next). At each stage of the search, each term is considered both for pruning and for expansion (if appropriately annotated with additional disjuncts). Simply, the term/strategy combination that yields the least non-zero cost is selected, so the precision of the remaining query is least affected.

5 Sources of Disjunction

As there are no true synonyms, the introduction of disjunctive elements into a query must necessarily reduce its precision. The decline is affected by a number of factors, such as the intrinsic ambiguity of a term (i.e., the number of different WordNet synsets it appears in [4]) and more importantly, the intrinsic ambiguity of each synonym. For example, a query can be seriously diminished by the introduction of a highly polysemous term into a query as a disjunctive alternative for a relatively homonymous question term (e.g., introducing “bug” as a synonym for “glitch”). The cost of expansion should thus reflect the potential decline in precision, and as such, should be a direct function of both the answer utility of the term being expanded, and of the total polysemy of the expansion set being introduced. The former quantifies the utility that can be lost in the expansion, while the latter essentially quantifies the number of ways in which it might be lost.

Recall that the enabling assumption of IR-based NLQA is that questions significantly resemble their answers. But the introduction of too many disjunctive forms into a query can result in answers that look nothing like the question. Furthermore, if such an answer also proves to be of low relevance, it may not be apparent to the user why it was generated, and users will simply not use systems they cannot trust. Expansion should thus be minimized, by introducing disjunction into a query in a gradual fashion so that when possible, answers can be retrieved using the original text of the question. Now, because all the possible disjuncts of a term are already present in a meta-query as annotations, the rate of expansion is completely governed by the structure of the query. A gradual deployment of synonym knowledge can be achieved by organizing the disjunctive set of a term into layers, so that the strongest synonyms are expanded first while more tenuous members require successive expansions to be revealed.

The disjunctive set of a term is constructed from a variety of conceptual relations, as derived from lexical knowledge sources like WordNet, a good source of synonyms, hypernyms and hyponyms. Statistical techniques can be also used to extract significant other terms from the English glosses that accompany each WordNet definition (e.g., the gloss of “surgeon” yields “surgery”). Yet perhaps the most useful knowledge source is the finite set of irregular verb forms, since the most common verbs are irregular (e.g., knowing that “spun” is a disjunct of “spin”, or vice versa, can be invaluable). These sources are considered in order of similarity when constructing the annotation structure of a query, so that the disjuncts least likely to diminish precision (such as irregular forms) will be considered for expansion first. The intent of course is to ensure that the least important terms are expanded before more focal ones, and that any expansions are done with the most reliable disjuncts first. Consider for instance the following ordering:

β: *Past, Past-of, Plural, Plural-of, Past-participle, Past-part-of, Nominal, etc.*

γ: *Synonym, Hyponym*

$\gamma\gamma$: *Hypernym, Partonym, Holonym*

The schedule is organized according to the utility of each relation type: the irregular forms are viewed as beta relations that introduce no loss of precision into a query, and are thus rewarded with a *#meta_beta* annotation that reflects their indifferent syntactic utility. In contrast, synonyms and hyponyms are viewed as potentially dangerous gamma-terms, while the lowest double-gamma rating is reserved for hypernyms, partonyms and holonyms. Note that the alpha annotation is not used given to any relation type on the schedule, since this is reserved for the original term as it appears in the question. This means that answers containing actual question terms are more valued than those that simply contain related disjuncts.

Figure 1 illustrates the annotated query structure generated from the question “Who was the *captain* of the *Titanic* when she was *struck* by an *iceberg* and *sank*”.

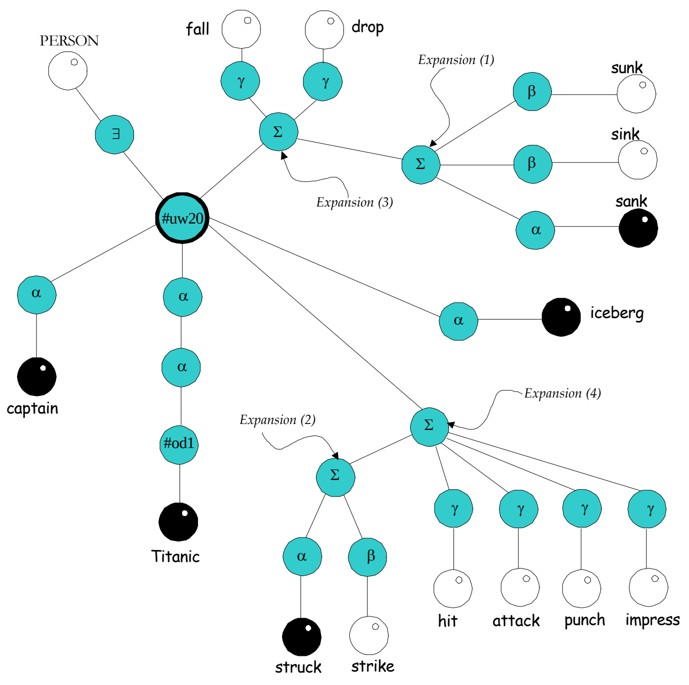

Fig. 1. The annotated structure of a meta-query. Annotations are shown using Greek letters, with Σ denoting *#meta_syn* and ∃ denoting *#meta_target*. The sequence of expansions is marked. White nodes depict annotation data, not query arguments, until an expansion occurs.

6 Evaluation and Concluding Remarks

Meta-knowledge annotations are used to maximize question throughput without diminishing answer quality. The approach was therefore evaluated in a performance comparison between CoreAnswer, a product of Coreintellect inc. that uses rich query annotations, and Lasso, the top-rated system in the 1999 TREC QA evaluation (in the 250-byte answer category, [5]). Lasso (licensed by Coreintellect under the name Arrow in 1999) uses the same IR framework for NLQA but employs a flat query formulation, compensating for this lack of structure by its willingness to retrieve and evaluate many hundreds of documents per question. Lasso achieved good answer quality on the Coreintellect archive of authoritative business content (around 5 million articles), as measured against a benchmark set of 100 business questions such as "How much did Peoplesoft pay for Vantive" (note that Lasso is being steadily improved, using techniques such as abductive text inference [6]). However, Lasso's large retrieval sets created considerable throughput problems, with many questions requiring over 30 seconds of CPU time. CoreAnswer was designed as an alternative that would maintain quality but drastically improve throughput.

On the same set of business questions, CoreAnswer produced answers that were equal or better in quality that Lasso for 92% of the questions. More significantly, CoreAnswer retrieved and analyzed an average of 23 documents per question, compared with an average of 617 documents for Lasso. Overall, CoreAnswer required an average of 3.7 seconds of CPU time per question, compared with 27.2 seconds per question for Lasso on the same platform. This increase in performance is clearly due to the dramatic decrease in retrieval size, with only a concomitant drop in quality of 8%. As such, we consider the meta-query approach to be a promising one, though considerable empirical analysis is still required to determine the full extent of the improvements that it offers.

References

1. Burke, R., Hammond, K., Kulykin, V., Lytinen, S., Tomuro, N., Schoenberg, S. (1997). Natural Language Processing in the FAQ Finder System: Results and Prospects. *Working Notes of the Spring AAAI Symposium on the World Web Web*, 1997.
2. Callan, J. P., Croft, W.B., Harding, S.M. (1992). The INQUERY Retrieval System. *In the proc. of the 3rd International Conference on Database and Expert Systems Applications*. 78 – 83.
3. Kwok, C., Etzioni, O., Weld, D. W. (2001). Scaling Question Answering to the Web. *ACM Transactions on Information Systems*, 19(3). 242-262.
4. Miller, A. G. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11).
5. Moldovan, D., Harabagiu, S., Pasça, M., Mihalcea, R., Goodrum, R., Girju, R., Rus, V. (1999). Lasso: a tool for surfing the answer net. *In the Processings of TREC-8*, 1999.
6. S. Harabagiu & S. Maiorano (1999). Finding Answers in large collections of texts: paragraph indexing + abductive inference. *Working Notes of the Fall AAAI Symposium on Question Answering*, Nov.1999.