The DARPA Machine Reading Program - Encouraging Linguistic and Reasoning Research with a Series of Reading Tasks[†]

Stephanie Strassel[1], Dan Adams[2], Henry Goldberg[3], Jonathan Herr[3], Ron Keesing[3], Daniel Oblinger[4], Heather Simpson[1], Robert Schrag[5], Jonathan Wright[1]

(1) Linguistic Data Consortium, (2) Scitor Corporation, (3) SAIC, (4) DARPA, (5) Global InfoTek, Inc.

Corresponding author: strassel@ldc.upenn.edu

Abstract

The goal of DARPA's Machine Reading (MR) program is nothing less than making the world's natural language corpora available for formal processing. Most text processing research has focused on locating mission-relevant text (information retrieval) and on techniques for enriching text by transforming it to other forms of text (translation, summarization) – always for use by humans. In contrast, MR will make knowledge contained in text available in forms that machines can use for automated processing. This will be done with little human intervention. Machines will learn to read from a few examples and they will read to learn what they need in order to answer questions or perform some reasoning task. Three independent Reading Teams are building universal text engines which will capture knowledge from naturally occurring text and transform it into the formal representations used by Artificial Intelligence. An Evaluation Team is selecting and annotating text corpora with task domain concepts, creating model reasoning systems with which the reading systems will interact, and establishing question-answer sets and evaluation protocols to measure progress toward this goal. We describe development of the MR evaluation framework, including test protocols, linguistic resources and technical infrastructure.

1. Background and Conceptual Framework

To advance research towards the end goal of general, lightly-trained systems which read text "in the wild" well enough to support automated reasoning tasks as well as to learn to read better, three independent Reading Teams are building universal text engines which will capture knowledge from naturally occurring text and transform it into the formal representations used by Artificial Intelligence. An Evaluation Team is selecting and annotating text corpora with task domain concepts, creating model reasoning systems with which the reading systems will interact, and establishing question-answer sets and evaluation protocols to measure progress toward this goal. While the Reading teams will be implementing a full range of natural language processing (NLP) techniques, significant emphases in the research will be in areas of machine learning and knowledge representation. Aligning a reading system's internal linguistic representation of a text corpus with the semantics of an external reasoning system – in fact, learning that alignment "on the fly" during reading – is a major challenge for the program. This paper describes development of the MR evaluation framework, including test protocols, linguistic resources and technical infrastructure.

The MR program is structured around a roadmap of linguistic and semantic capabilities, e.g. dealing with anaphora, causal and modal language, temporal and spatial reasoning, sentiment and belief. Over the course of the five-year program, the Evaluation Team will provide a series of graded Reading Tasks (described below), which present the reading systems with increasingly difficult challenges. At each phase, increasingly complex linguistic and reasoning tasks are combined with increased performance expectations in query answering, expanded corpus volume, and reduced time to prepare and adapt the systems to new tasks.

2. Readability Task

The program begins with an interesting but somewhat orthogonal challenge: assessing the "readability" or quality of a text passage. This is motivated by the belief that a computer system with the ability to extract a set of language features diverse enough and of high enough quality to assess readability at least as well as humans will be primed to take on the challenges of machine reading.

Measures of readability have been proposed and are in common use by educators and editors to estimate grade level or comprehension. These measures are usually based on surface features, such as sentence length or syllable count (Flesch, 1948; Kincaid, et.al., 1975), although some incorporate deeper concepts, such as word or sentence "complexity" (Gunning, 1952). However, recent research (Pitler & Nenkova, 2008) has shown that surface features are not well correlated with judgments by adult readers, whereas several syntactic, semantic, and discourse features are.

Reflecting the goal of the MR program, we define readability as a subjective judgment of *how easily a reader can extract the information the writer or speaker intended to convey*. We draw texts from a diverse range of genres which a machine reading system is likely to employ: newswire stories, weblogs, newsgroups/forum posts, Wikipedia entries, broadcast transcripts and closed caption text, and even some machine translation output.

To separate the rating of texts from the measurement of human performance, we employ two panels. A panel of judges with expertise in reading tasks such machine translation post-editing produces a gold standard judgment – a rating from 1 to 5 for each passage – and their mean defines a reference rating for that passage. A separate, "novice" panel of typical readers of English then rates the passages, to estimate the variable performance of humans

at this task. Machine performance is expected to meet or exceed the performance of individual novice readers with statistical significance.

2.1 Data and Human Judgments

Genre and passage selection and human test protocols are designed to ensure minimal bias from such factors as fatigue, topic familiarity, genre-specific surface features, and inter-passage ranking interference. Individual reading passages are selected by first preparing a large pool of candidate documents for each genre. Candidates are manually vetted to exclude inappropriate passages (e.g. not in English); the pool is then automatically downsampled to produce the targeted number of passages for each genre. Selected passages are processed to standardize formatting and remove any genre-specific surface features like newswire headlines or Wikipedia-style markup. Passages are also assigned to a general topic category (current events, sports, etc.) prior to assessment.

The prepared passages are then assigned to expert and naive judges for assessment via a web-based user interface. To reduce assessor fatigue, passages are grouped into sets of 10, called rounds. For each round, assessors first read each passage and give a rating of 1-5 "stars" to indicate how readable the passage is. After rating all passages in a round, assessors then provide a rank ordering of those passages in terms of their overall readability. Following each round expert judges are also asked to state the criteria they used to determine the passage ratings/rankings. This is an open list, not a set of pre-supplied criteria. Assessors continue rating, ranking and (experts-only) listing criteria for each round until all passages have been judged. All passages are assessed by multiple independent naive and expert judges. Presentation order within and across rounds is randomized, and each round is roughly balanced for genre and topic. A similar balance is maintained between training and testing data sets.

2.2 Evaluation, Metrics, and Results

To reduce the chance that extraneous factors might dominate the test, such as variability in the expert ratings or bias from the experts' experience with previous linguistic annotation tasks, we devised several scoring metrics with different statistical characteristics to compare machine to (novice) human judgments.

Metric 1 – Score Difference. This metric measures how much closer than the average novice the machine comes to the gold standard rating. We use the mean of the expert panel ratings as the reference score *s(g,t)* for text *t*. Let *s(j,t)* be the score of the j^{th} novice judge. Then

$$\bar{\Delta}(humans,t) = \frac{1}{N}\sum_{j=1}^{N} abs(s(g,t) - s(j,t))$$

is the mean delta over all novice judges on text *t*, and the machine delta is $\Delta(m,t) = abs(s(g,t) - s(m,t))$.

$$\bar{\Delta}(humans,m) = \frac{1}{k}\sum_{t=1}^{k}\left(\bar{\Delta}(humans,t) - \Delta(m,t)\right)$$

is the mean difference between the machine and the average novice estimate of the gold standard for each text.

Metric 2 – Proportional Target. To account for variation in the expert judgments, we define a target for each passage as the range of expert ratings, e_i, and award a score inversely proportional to target width. The metric is simply $\frac{1}{k}\sum_{t=1}^{k} hit(m,t)$

where *e(i,t)* is the judgment of expert *i* on text *t;*
and hit(m,t) = 1/(1+max(e(i,t))-min(e(i,t)))

Metric 3 – Correlation Coefficient. The claim of the readability evaluation is that machine ratings of texts are closer to expert ratings than are novice ratings. A particularly simple and robust way to test this claim is to ask whether the correlation between machine scores and expert scores is higher than you would expect if the machine is no different than a novice.

We use the expert panel mean to define the gold standard expert rating and compute, as our test statistic, the Pearson correlation coefficient of this value and the machine score for all passages in the set:

$$\rho(E,M) = \frac{\mathrm{cov}(EM)}{\sigma_E\sigma_M} = \frac{\sum_{t=1}^{k}(e_t - \mu_E)(m_t - \mu_M)}{(k-1)\sigma_E\sigma_M}$$

where E={e_t | *t=1…k*} are the gold standard, and M={m_t | *t=1…k}* are the machine scores.

Evaluation. To verify that the machine's performance is significantly better than novices', we derive a sampling distribution under the null hypothesis that machines and novice humans perform identically by repeatedly choosing a random novice score for the machine's score. We typically generate 10,000 iterations to ensure stability of the sampling distribution parameters. In each metric, the upper critical value (2.5% tail) is used to demonstrate 95% confidence.

Results. Preliminary results on both cross validated and blind test passages by three separate reading systems are very promising. Table 1 shows p-values (the estimated likelihood that metric statistics as good as the machines' would be produced by novices).

p-value	System A	System B	System C
Metric 1	0.0002	0.0002	0.0001
Metric 2	0.0002	0.0002	0.0001
Metric 3	0.0002	0.0132	0.0001

Table 1: Preliminary results of Readability test

The Reading teams have reported using a broad mix of features, from "traditional" readability features like sentence length, to deeper, more reading-relevant features like parse complexity, verb features, and number of extracted assertions. They confirm Pitler & Nenkova's (2008) finding that surface features are not sufficient by themselves to succeed at this task.

3. Reading Tasks

The fundamental challenge for reading systems is to extract knowledge from natural language texts into formal statements in a focused ontology to support a performance task in a particular domain (a *reading task domain*). The Machine Reading program defines a crisp boundary between a reading system and a *domain-specific reasoning system* (DSRS) that captures task-specific background

knowledge that either:

- Would not usually be found in the target texts; or
- Would be considered out of scope for machine reading during a given (e.g., early) evaluation period.

A reading system can invoke the DSRS to determine:

- Whether a given set of formal statements is internally consistent; and
- What other, output formal statements follow logically and/or probabilistically from a given set of input statements.

The reading system can exploit this information to help resolve linguistic ambiguities and textual contexts. Simply parsing all texts into a single, large, undifferentiated set of statements and then making all possible inferences would likely yield hopelessly inconsistent and useless results, and the DSRS thus helps the reading system meet the key challenge of determining which statements should be used together in distinct contexts.

Reading tasks are motivated by real-world tasks that can productively exploit the results of reading and inference. The program plans to take on two new task domains in each program year. Within a particular task domain (e.g., international political analysis, equipment maintenance, medical diagnosis), we devise a series of *use cases* reflecting a progression of complexity or sophistication in natural language understanding and in formal knowledge representation.

A concrete example will serve to illustrate the reading task framework. In an early Phase 1 reading domain, *NFL Scoring*, the task is to read news articles about National Football League games and answer queries about final scores, scoring events, and periods played. This domain was selected considering the motivating performance task, "Dynamically infer and display status information regarding a sports competition from a play-by-play account," and the notional application, "For the sports junkie, a set-top plug-in reads the closed captioning stream and maintains more comprehensive status than the mini-scoreboard overlay typically used in broadcast video, such as a summary of the game's scoring events."

Note that this task engenders inference: Suppose the announcer says, "The kick is good, and it's 6 to 3," and suppose the only earlier score extracted was "Atlanta Falcons 3, New York Jets 0." A domain-specific reasoning system (DSRS) can infer that the "kick" must have been a field goal, not a point after touchdown; it may need more information (e.g., a future score) to determine which team is ahead at this point. For simplicity early in the program, we have adjusted this performance task so that we can work with news stories rather than play-by-play transcripts: Given any score in an NFL game and any partial information about the contributing scoring events, the DSRS will return the most likely scoring event combinations and their probabilities. Ultimately, we target reading systems capable of exploiting such probabilistic information from DSRS results during the reading process (e.g., to determine the most likely grouping of extracted formal statements into consistent contexts).

3.1 Reading Task Resources

Associated with each reading task, the evaluation team prepares a package of linguistic and knowledge representation and reasoning resources, including the following.

- A *text corpus* for reading system training and a similar one (revealed only during an official evaluation period) for testing. For NFL Scoring, we have an initial training corpus of some 100 news stories.
- A *syntax specification* expressing the expected formal outputs of machine reading and expected formal inputs to the DSRS. In principle, this specification could require any formalism; in initial practice, the evaluation team provides an RDF/OWL *ontology* (RDF, 1994; OWL, 2004) specifying classes and properties (entity types and relations) and associated semantic constraints (e.g., type and cardinality restrctions on properties). An excerpt of the NFL Scoring ontology appears in Table 2. Note that besides elements (e.g., NFLGame, gameDate) that are designated as direct targets of machine reading, the ontology includes properties (e.g., touchdownCompleteCount) that the DSRS will either accommodate as input or infer and built-in properties that can appear only in queries or the bodies of rules.

Class (Superclass…)		
property	(Type)	*cardinality*
NFLGame (Event)		
gameDate	(Date)	# ≤ 1
numberOfPeriods	(Count)	# ≤ 1
hasOvertimePlay	(Boolean)	# ≤ 1
gameWinner	(NFLTeam)	# ≤ 1
gameLoser	(NFLTeam)	# ≤ 1
homeTeamInGame	(NFLTeam)	# ≤ 1
awayTeamInGame	(NFLTeam)	# ≤ 1
teamInGame	(NFLTeam)	# ≤ 2
teamFinalScoreInGame	(TeamSummaryScore)	# ≤ 2
TeamSummaryScore ()		
teamScoringAll	(NFLTeam)	# = 1
pointsScored	(Count)	# = 1
touchdownCompleteCount	(Count)	# ≤ 1
onePointConversionCompleteCount	(Count)	# ≤ 1
twoPointConversionCompleteCount	(Count)	# ≤ 1
fieldGoalCompleteCount	(Count)	# ≤ 1
safetyCompleteCount	(Count)	# ≤ 1

Table 2: NFL Scoring Ontology Excerpt

- *Annotated examples* of reading target formal statements manually identified in the training corpus.
- *Annotation guidelines* that provide expectations both for corpus annotators and for reading teams and that set forth ground rules to cover anticipated borderline cases.
- A set of task-relevant formal *queries* that a reading system (supported by the DSRS) must answer. The evaluation team provides an initial set of queries with the training corpus. The ontology serves as a fair-game basis for additional or alternative queries during testing.
- *Formal answers* to the sample queries. These answers constitute the *gold standard* for the reading task (not to be confused with that for the readability task described in Section 2).
- A DSRS.

Table 3 illustrates inter-relationships among queries, DSRS inference rules, and input RDF statements extracted by reading systems or humans, using an example from the NFL Scoring reading task. Note that individual RDF statements are notated as subject-predicate-object triples and RDF graphs (collections of RDF statements) as parenthesized lists.

Query Graph *"How many periods were played in each game mentioned?"* ((?G type NFLGame) (?G numberOfPeriods ?N))
Inference Rule 1 *"If overtime is mentioned but no period count, assume 5."* (:if (?g hasOvertimePlay true) :if-not (?g numberOfPeriods ?n) :then (?g numberOfPeriods 5))
Inference Rule 2 *"A statement of overtime play contradicts one of ≤ 4 periods."* (:if (?game numberOfPeriods ?N) (?N mrq:lessThanOrEqualTo 4) (?game hasOvertimePlay true) :then (?inputGraph contradicts NFLScoringTheory))
Input Statement Graph *(Statements extracted by reading or by annotation.)* ((Game-1 type NFLGame) (Game-1 hasOvertimePlay true))
Answer Graph *"Five periods played." (Inferred by invoking the DSRS with statements extracted from text.)* ((Game-1 type NFLGame) (Game-1 numberOfPeriods 5))

Table 3: Elements of the NFL Scoring reading task. A reading system calls the DSRS, providing query and input statement graphs. The DSRS exercises its inference rules and returns one or more answer graphs.

A query is an RDF graph containing variables (e.g., ?G, ?N); an answer is an instantiation of the query that binds values to the its variables in a way that is consistent with domain theory embodied in the DSRS' rules and with the input statements extracted from the text corpus.

In this example, DSRS Inference Rule 1 provides the (default) assumption that overtime results in five periods of play. In fact, a post-season NFL play-off may have multiple overtime periods (if the score remains tied). Instead of a purely logical default, we could also have exercised probabilistic reasoning here and returned an answer with a probability less than 1.0. Either way, the reading system will learn a likely number of periods, making up for the fact that most stories do not comment on any lack of overtime play (which is exceptional) or name the number of the final period, even when overtime happens (as second overtimes are quite rare). DSRS Inference Rule 2 states that NFL games are limited to four periods before any overtime period. This rule does not fire given our example input, which is consistent with the domain theory.

Note that the evaluation team employs the DSRS in its process of generating sample answers to the training queries for the training corpus. We feed the input statement graph resulting from the formal annotation of a given news story into the DSRS. The DSRS will detect any domain theory contradictions in this process, which the annotation team can then repair to improve the annotated corpus. The DSRS thus serves in gold standard answer development and in annotation quality assurance, as well as in reading system support.

The DSRS' quality assurance of annotations is facilitated by its tracking and perspicuously (for human users) displaying two kinds of *provenance* (or warrant), for statements that either are input to or are derived by its inference process. Input statements are supported by *text-based provenance* that associate specific text excerpts with a statement and its subject, predicate, and object. Text-based provenance is created by the annotation tool (see Section 3.2.2) when annotators link formal statements and their parts to segments of a document's text via the GUI. Text-based provenances themselves are expressed using RDF/XML statements; the evaluation team reviews these via a stylesheet-enabled HTML browser view to validate annotations' usage of the reading task ontology. DSRS-derived statements are supported by *rule-based provenance* that link applied rules' specific consequents (instantiated "then" parts) with their associated specific antecedents (instantiated "if" and "if-not" parts), where each antecedent is instantiated by matching it to a DSRS-input or earlier-inferred statement. Users thus may trace back through connected rule-based provenances to traverse the derivation tree supporting any inferred statement. Rule-based provenances also are expressed as RDF/XML; their review is supported by a similar stylesheet-enabled browser view. By reviewing rule-based provenances for any contradiction rules (e.g., Inference Rule 2 in Table 3) that fire during gold standard answer generation, the evaluation team can determine what semantic issues require our attention and correction. Reading systems can similarly (presumably, automatically) employ rule-based provenance to debug any DSRS-detected inconsistencies in their inference input graphs.

To help address the extremely challenging rapid prototyping-and-production development issues that we face in providing resources to facilitate the coupled advance of both linguistic and knowledge representation technologies (discussed further in Section 3.2.2), we are exploring a tighter integration of our underlying tools (i.e., the DSRS and the annotation tool's RDF generation capability) and their user interfaces to expedite the delivery of quality assurance feedback to both knowledge representation and annotation specialists on the evaluation team.

3.2 Annotated Corpora

3.2.1. Approach

Source text is selected for each use case to provide content-appropriate material for training and testing. In early phases the training and test corpora are small -- on the order of 100 documents -- but will grow exponentially over the program phases, reaching web scale in Phase 5. Some use cases may also include a large background corpus to support unsupervised statistical learning methods.

There are three annotation components for each Reading Task. The first is a syntax mapping, consisting of naturally-occurring text examples of the formal assertions con-

tained in the syntax specification. The syntax specification and syntax mapping together form a set of example pairs linking formal assertions to sentences or phrases with the same meaning as those assertions. The second annotation component is the set of answers to Reading Task domain training queries extracted from naturally occurring text; the third component is the set of answers to test queries.

Manual annotation strategies also vary with each use case. While some use cases require limited, incomplete and/or query-driven annotation, in the case of NFL Scoring near-complete annotation of the corpus was used to simultaneously extract syntax mapping examples and training query answers. Annotators read each document and labeled all ontological categories occurring in the text. The full annotation approach was well-suited to this initial use case for several reasons: the relatively small corpus size permitted a streamlined annotation workflow with limited human effort; it did not rely on user-driven search to find answers, which could have led to gaps in annotation coverage; and, since queries did not need to be defined prior to annotation, it allowed the final selection of training queries to be driven by answers actually occurring in the text.

While complete annotation for training data was appropriate for NFL Scoring, query-driven annotation is needed in other cases. Reading System evaluation is based on comparisons to human performance, so for test queries the human annotation workflow must closely parallel the machine task. Formally specified queries submitted to Reading Systems will be re-formulated as natural language questions for human readers. Both humans and Reading Systems will generate query answers that include text provenance from the data source. Humans may produce multiple overlapping response assertions; the combined set of answers will be manually adjudicated to determine which answers are correct, and which if any are out-of-scope for that phase. The correct, in-scope adjudicated human responses become the gold standard answer set for that query.

3.2.2. Annotation Infrastructure

Creating annotated corpora for the Reading Tasks presents a number of technical challenges, foremost among them the alignment between natural text and ontological categories. An ontology representing the semantics of some domain has its own requirements and characteristics divorced from the natural language expression of those semantics. Good design demands that an ontology be modular, unambiguous, and consistent in its representations; natural language is often none of these things. Consider the so-called NFLGame entity, part of the NFL Scoring use case. A typical NFL news report involves a number of entities and events mentioned in the context of one or more particular games, but the report may lack any explicit mention (naming) of the games in question, because the typical human reader implicitly understands that mentions of teams winning, losing or scoring points occur in the context of a particular game event. The ontological representation of these sub-game scoring events ties them together, through an intermediate object called NFLGameTeamSummaryScore, to a game object, called NFLGame. If text provenance can be found for NFLGame, it is directly mapped to that object, but the ontological object is created regardless of its text instantiation (or lack thereof). The intermediate NFLGameTeamSummaryScore object does not map directly to text at all, so is never given text provenance. Therefore, while ontological and linguistic expressions can and do come together in many places, it was recognized early on that direct, transparent mapping of ontological categories to naturally occurring text was not an appropriate task for human annotation due to the general mismatch of surface versus formal representation.

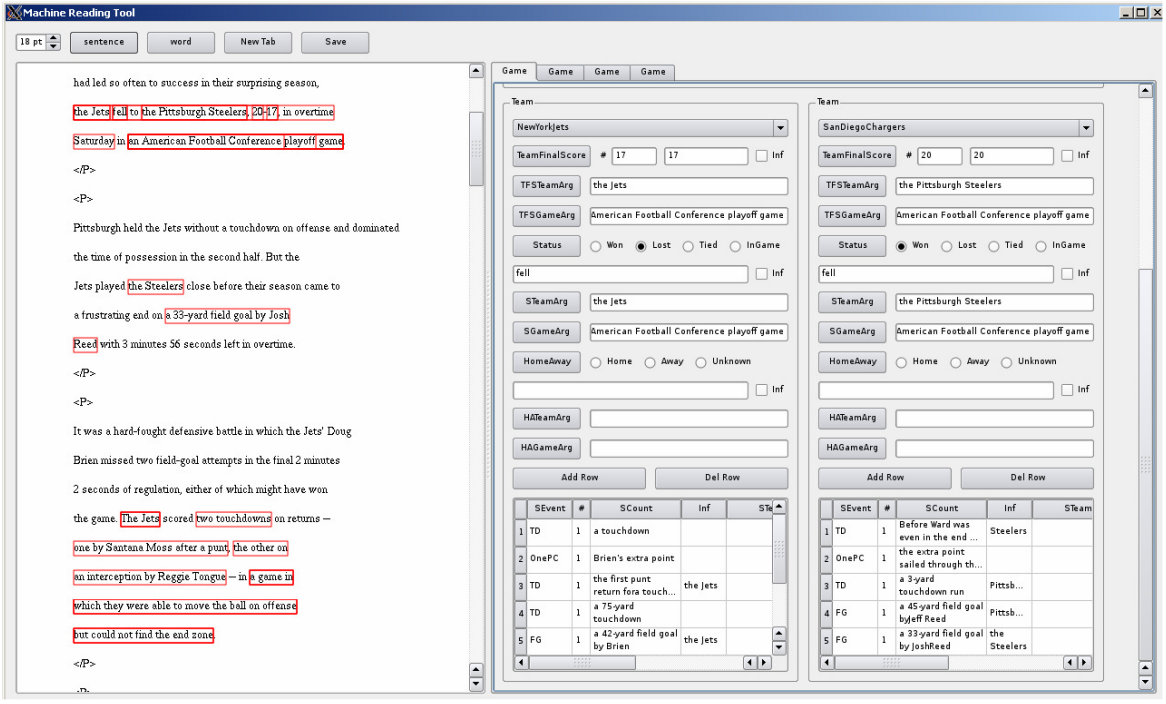

Figure 1: Screenshot of Annotation GUI, Configured for NFLScoring Use Case

To address this issue and make annotation a practical task for humans, the formal assertions of the ontology must be recast into annotation categories that more closely capture the patterns found naturally occurring text. A careful mapping between ontology and annotation categories must take place for each new use case. During annotation, the human does not interact directly with the formal ontology, but utilizes only the annotation categories, mediated by a customized annotation GUI and annotation guidelines. The annotation backend therefore must maintain a direct mapping between labeled instances of these categories in the corpus and the output of formally specified ontological assertions.

Creating an accurate and complete mapping between the annotation categories and the formal ontology is challenging, particularly in an environment of multiple, independent, and ever-evolving use cases. This frequently changing landscape requires more than rapid prototyping of software, but also rapid deployment of final products (i.e. annotated corpora and the tools required to create them). Annotation GUI functionality may need to be extended quickly to address new requirements, while annotation categories and labeling guidelines are still evolving to reflect the latest ontology. Moreover, with the goal of progressively more difficult reading tasks over the life of the program, human annotation requirements are also expected to become more challenging and complex over time, which requires a robust and extensible annotation infrastructure.

The solution to these technical challenges is the development of a single, but highly customizable, annotation toolkit, as illustrated in Figure 1. The functionality and primary code base of the tool increases in stability with each cycle, because the inherent variability of the use cases is externalized to a configuration file. Both the annotation categories and the subsequently produced ontological categories are defined in the configuration file, in a domain specific language.[1] The annotation tool interprets the configuration file to determine how to lay out the widgets in the GUI; the file also specifies how annotation output should be structured. In essence, the code that necessarily changes from use case to use case is represented in a separate file and in a highly simplified and constrained form that minimizes development time for new use cases. These substantial technical challenges have been hidden from the annotator by reformulating the annotation task from the base requirements.

3.3 Evaluation Protocol and Metrics

The core performance task for MR is answering queries posed by an automated reasoning system, so performance evaluation is focused on determining the correctness and completeness of query answers found by Reading Systems. An additional goal of Machine Reading, beyond accuracy, is *generality*; in later phases of the program, time and effort required to prepare for "blind" use cases or entire domains will be measured as well.

[1] "Domain" here does not refer to the reading task, and in fact the domain specific language does not change from one use case to another.

The set of correct answers for a given query – the gold standard answer set – is determined collaboratively based on answers found by humans and machines and adjudicated by experts, as in the TREC and TAC programs. Each phase of the program includes specific goals for reading and reasoning capabilities that are considered "in bounds", and only answers that can be found within the range of phase-specific capabilities are included within the gold standard for that phase.

Since the goal of the Machine Reading program is to approach human performance, the primary query answering metric is the ratio of machine to human F measures (the – possibly weighted – harmonic mean of precision and recall) over the gold standard answer set.

The Machine Reading program distinguishes between the ability to find answers that are stated explicitly in text – facts – and the ability to find answers that require inference. Separate F-measure performance ratios are computed for answers stated directly in text (PR_F) and those requiring inference (PR_I).

This distinction between fact and inference – along with the development of a consensus answer set – raises a series of interesting challenges. For example, how do we arrive at an adequate set of queries and gold standard answers which are consistent with the ontology and based on consistent human annotation? As mentioned above, queries are merely incomplete assertions consistent with the task ontology. We are able to generate the full set of possible queries from the ontology, so depending upon the size of that set, all or a portion of these queries are automatically applied to the annotations. The human annotations – also consistent with the ontology, as described in Section 3.2.2 – provide an initial set of gold standard answers to queries involving the classes and relations they involve. So, in the example we've been using, the relation numberOfPeriods is annotated in several texts. Specific, annotated games and numbers which fit the query template are available as a sub-set of all answers to the queries. The remaining answers come from two additional processes. First we take the directly annotated assertions and run them through the DSRS rules. This would produce, for example, an inferred answer of 5 periods for a game where a mention of overtime was annotated. This answer is added to the gold standard set. Then, during testing, all answers provided by reading systems are retained by the evaluation framework. These may include correct answers which should be included in the gold standard. We have designed an adjudication process where annotators can quickly view the machine-provided answers along with their provenance (portions of the text where the answer was read, or where precursor facts were read from which the answer was inferred).

Reading systems, indeed all readers, perform inference at many levels, from resolving co-reference to applying common sense reasoning or learning about the domain through reading the corpus. To make the notion of extracting explicitly stated facts versus inferred assertions more precise, we instruct annotators to extract explicit statements of the targeted assertions. We classify these as "explicit" and the assertions derived from applying the DSRS rules over these are "inferred". When additional

assertions are found by reading systems, the same distinction is made by the adjudicator – is the assertion explicitly stated in the piece of text which the reading system returns as provenance, or are more than one, disjoint pieces required to support the assertion? In the former case, the answer is scored as a fact and included within the gold standard for fact-based answers, while in the latter it is scored as an inference.

Another challenge for evaluation is scoring when complete enumerations of all entities are not available. In the simple domain of NFL Football, we have fully annotated the corpus and thus enumerated all teams and games, but it is clearly impossible in broader domains, such as news about political protests, attacks, or other emerging events to perform a similar enumeration. We have designed the machine-test platform interface to include specification of an "equivalence" class for each entity returned by a machine reader. These classes (sets of referenced strings) are matched against the gold standard, allowing for partial credit as well as adjudication and improvement of the gold standard.

3.4 Infrastructure

One of the goals of the machine reading program is to provide a platform around which a community of machine reading experts can grow. In order to facilitate that future we chose a design for the platform and system architecture which relied upon open standards that would make the transition from DARPA program to community project as seamless as possible. Open standards and a SOA design also mean that reacting to inevitable change can be

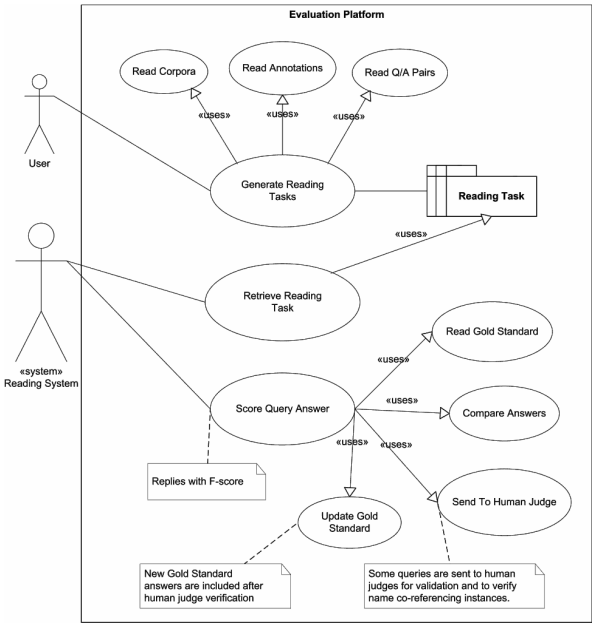

Figure 2: Architecture of the Machine Reading Evaluation

achieved with only minor impacts to users. A second, and more immediate goal for the platform is that it provide a means for the program participants to get early access to the evaluation phase interfaces, file formats and data types. In order to ensure the platform is available at all times and will run in a variety of environments we created a version of the platform which provides all evaluation phase interfaces and a subset of the functionality as a set of Java™ web services. Providing all evaluation phase interfaces early in the life of the program provides us with time to receive and incorporate feedback. Most importantly, it also gives the program participants time to become comfortable with the evaluation platform by running their own dry-run tests as early and as often as is useful to them.

As illustrated in Figure 2, the platform supports three primary use cases which we'll discuss. These include reading task generation, reading task consumption, and query answering facilities.

A reading system can make use of the reading task generation functionality to create complete reading tasks similar to those which will be used at evaluation time, or to create reading task sub-sets which focus on specific areas of interest. Once generated, a reading task is available for consumption by the reading system.

During a formal evaluation reading task generation will be performed by the evaluation team to ensure they have control over the contents. Therefore, a reading system will be given the identifier of the reading task they are being evaluated on so that it can be retrieved using the reading task retrieval interface. This interface simply streams the reading task RDF graph to the reading system via either a SOAP or a REST protocol web service.

Prior to a formal evaluation a reading system will want to gauge its progress on the upcoming task. To facilitate this need the platform provides an automatic scoring system which compares reading system answers to a set of gold standard answers compiled by annotators for the set of queries in the reading task. The automatic scoring process is imperfect because the human judge's which are used during an evaluation are not present to verify that answers are absolutely correct. However it does provide a baseline from which progress can be measured. During scoring, detailed logs are kept of answers provided by each reading system. These are gathered by the evaluation team periodically for review. Answers found by the reading systems which are deemed to be valid but missing from the gold standard are added. In this way, over time, the gold standard for a reading task becomes increasingly complete.

4. Conclusion

While the first year's metrics will not involve specific task performance goals, the program is based on a series of increasingly challenging Q/A tasks.

The Evaluation team will produce over 30 use cases in at least 10 task domains over the life of the program. We believe this series of graded, diverse reading tasks, comprising well annotated texts paired with model reasoning systems, interface knowledge in machine-accessible form, and gold-standard Q/A pairs tied to significant reading tasks, will be a significant resource for NLP researchers far beyond the program itself.

An important goal of the program is to encourage research beneficial to machine reading by making these materials available to the research community at large. The resources described within this paper will be made available to the broader research community over time. Many re-

sources will be distributed to LDC members and non-member licensees through the usual methods, including publication in LDC's catalog, while other resources will be freely distributed without licensing constraints.

5. References

Flesch, R. (1948). A new readability yardstick, Journal of Applied Psychology, Vol. 32, pp. 221–233.

Gunning, R. (1952). *The technique of clear writing*; McGraw-Hill International Book Co; New York, NY.

Kincaid, J. P.; Fishburne, R. P., Jr.; Rogers, R. L.; and Chissom, B. S (1975). Derivation of new readability formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy enlisted personnel, Research Branch Report 8-75, Millington, TN: Naval Technical Training, U. S. Naval Air Station, Memphis, TN.

OWL - Web Ontology Language Overview (2004). World Wide Web Consortium, http://www.w3.org/TR/owl-features.

Pitler, E. and Nenkova, A. (2008). Revisiting Readability: A Unified Framework for Predicting Text Quality. Proceedings of EMNLP.

Proceedings of the First Text Analysis Conference (TAC 2008), Gaithersburg, MD, Nov. 2008. National Institute of Standards and Technology.

RDF - Semantic Web Standards (1994). World Wide Web Consortium, http://www.w3.org/RDF.

Voorhees, E. M. (2005). TREC: Experiment and Evaluation in Information Retrieval. MIT Press.

[†] The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

Approved for Public Release, Distribution Unlimited.