A Unified Representation and Inference Paradigm for Natural Language Processing

Dan Roth and Mark Sammons

University of Illinois at Urbana-Champaign
{danr|mssammon}@uiuc.edu

Abstract

Traditional approaches to Natural Language Text Processing limit performance and flexibility by committing to canonical reprentations of input text, while many NLP applications for general tasks such as Textual Entailment use ad-hoc architectures with limited flexibility, and which limit the expressiveness of inference procedures over components. We present a Modular Representation and Comparison Scheme (MRCS) that addresses these problems by combining a modular representation with a modular, unification-like inference algorithm that allows the system architect to defer appropriate disambiguation decisions until run-time.

1 Introduction

General Natural Language Text Processing applications tend to suffer from two problems: overcommitment to a particular representation of input text, and ad-hoc architectures that limit the ability to incorporate new resources, and which may also limit the flexibility of their inference procedures.

Traditional Knowledge Representation and Reasoning approaches in (Moore, 1986; Schubert, 1986; Hobbs et al., 1988; Blackburn et al., 1998) aim to determine a (set of) *canonical* representation(s) of text, represented in some formal language, in order to manipulate them later and support inference with the aid of Knowledge Bases encoded at the level of their formal representation. Such approaches require, for each span of input text, multiple decisions about ambiguous terms, and puts the entire processing/disambiguation burden on the representation induction step. More recent approaches, recognizing the inherent problems of such early commitments, either maintain multiple logical representations (Crouch, 2005) or use semantic underspecification to avoid early commitment to certain kinds of decisions (Pinkal, 1999), particularly scopal ambiguity and lexical polysemy. However, they still depend heavily on the accuracy of automated processing of the input.

Recent advances in some NLP tasks, such as the context-sensitive verb paraphraser of (Connor and Roth, 2007), aggravate the problems associated with canonical representations. This resource takes as input a sentence with a marked verb or verb phrase, and a candidate verb for replacement, and indicates whether the candidate is a valid replacement. Such resources make little sense when applied to a question or text span in isolation: on what basis can candidate substitutes be selected? But in the context of RTE and QA, verbs or verb phrases from candidate Hypotheses/answers can be used by this module as candidates to determine whether they are matched by a verb or verb phrase in the Text (or vice versa).

Challenges such as the TREC Question Answering (QA) track (Dang et al., 2007) and The PASCAL Recognizing Textual Entailment (RTE) challenges (Dagan et al., 2006) have provided an arena in which to test general NLP techniques. Solutions typically combine a range of NLP resources, but in apparently ad-hoc ways; it is not clear, given a new solution for a subtask like verb paraphrasing, how they can be integrated.

In this paper we argue for a dual architecture of Representation and Disambiguation/Comparison functionalities capable of leveraging a range of NLP subtask solutions, while avoiding some of the problems of traditional approaches. The representation aspect of the architecture supports multiple analysis views linked flexibly to support inference that cuts across views, while the inference functionalities, employing "expertise" in different analysis, are controlled by an application level inference procedure. To this end, we present a Modular Representation and Comparison Scheme (MRCS), an intermediate, non-canonicalizing representation and inference paradigm appropriate for many NLP applications, including Question Answering and Textual Entailment.

2 Motivation

Recognizing Textual Entailment (RTE or just TE), as formulated in the PASCAL RTE Challenges (Dagan et al., 2006), is the task of identifying whether, given two spans of text, a human reader would agree that the meaning of the second span is entailed by the meaning of the first. As such, a solution for RTE could be used in a wide range of NLP applications, such as Question Answering (by representing the question as a natural language statement and comparing it to candidate answers) and Information Extraction (by representing relations of interest as natural language text and comparing it to text spans that might contain such relations). As such, it is a good representative NLP task with which to evaluate NLP solutions.

To ground our argument for delayed disambiguation and the approach we use to accommodate it, we use the Textual Entailment example presented in Figure 1. TE examples comprise two text spans, a Text and a Hypothesis; the task is to determine whether a human reader would believe the Hypothesis is true given the Text and some undefined set of World Knowledge.

In the figure, the representations for the Text and Hypothesis text spans contain multiple overlaid views, each representing analysis by a separate automated source. SRL (Semantic Role Labeling) Constituents are connected by Relations that indicate composition structure, while Named Entity and Coreference Constituents have no additional structure. Sufficient information to determine that the Text entails the Hypothesis has been provided in these views (though many other levels of analysis are possible and desirable – for example, full or dependency parse structure, or logical forms corresponding to part or all of the text, which can be recorded in a similar way).

To avoid the problem of early commitments, it is desirable to a) defer such disambiguation decisions as may profit from additional contextual information until the comparison step, and b) maintain the original representation so that reinterpretation is possible. This requires an important assumption: that candidate Texts in RTE (answers in QA), even incorrect ones, can provide valuable context to resolve ambiguities in the Hypothesis (question) – and even, that such ambiguities may not need to be directly resolved at all.

To see why this is the case, consider the following RTE example:

Text: John was angry at Dick Jones, the man who killed his project.
Hypothesis 1: A man killed Dick Jones.
Hypothesis 2: Dick Jones was responsible for ending the project.
Hypothesis 3: Dick Jones put an end to the project.

Processing the Text in isolation and deriving a logical form requires a number of disambiguation decisions that depend on the KB resources to be applied: determining, for example, that "Dick Jones" is a named entity of type Person, and that the correct sense of the verb "kill" is here "stop" or "end". The Hypotheses must then undergo the same processing steps, with similar disambiguation problems. At the Comparison step, if the wrong sense of "kill" has been determined for the Text, the chance of correctly identifying Hypothesis 2 and 3 as being entailed is significantly reduced, while that of incorrectly classifying Hypothesis 1 as being entailed is significantly increased. Suppose now that in the Text, "Dick Jones" was not identified as a Named Entity, but that in the Hypotheses, it was. Handling this within the canonicalized logical approach is problematic, requiring the system architect to essentially model the kinds of mistakes made by the logical form inference process.

A delayed disambiguation approach can leverage the fact that the text span "Dick Jones" was labeled as a Named Entity in one of the two text fragments – even in an incorrect Hypothesis. It avoids the need to ever determine the correct sense of "kill"; the necessary disambiguation comes by virtue of the arguments of the verb in each Hypothesis. If the arguments match, and the verbs share a common sense, it is highly likely that the shared sense is the correct one.

MRCS is a suitable paradigm for NLP applications with these characteristics, maintaining original information in a way that allows re-interpretation based on application-time input, and which reserves more uncertain disambigua-

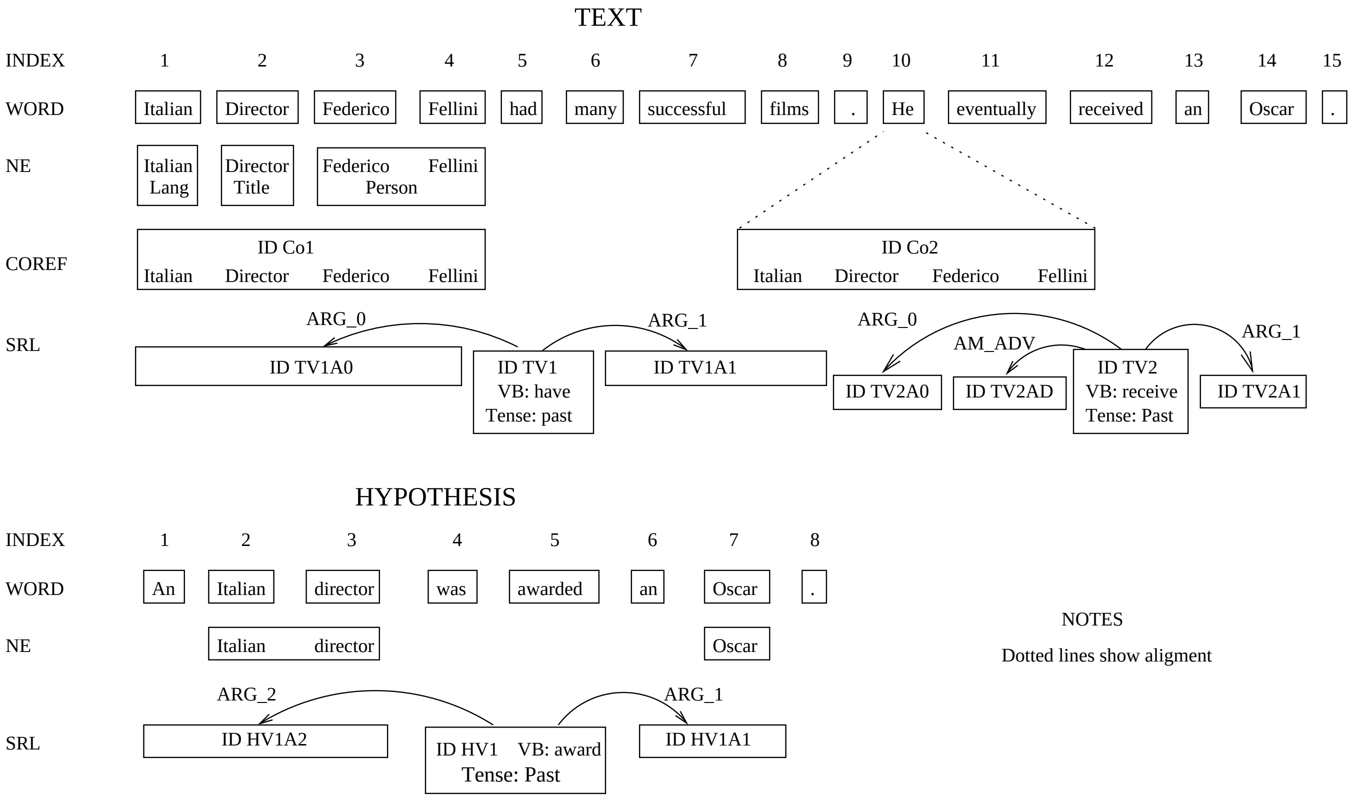

Figure 1: A Textual Entailment Pair as represented in MRCS.

tion decisions until application time. It thereby avoids over-commitment to some (set of) interpretation(s) of the input and/or over-generation of meanings, while accommodating Knowledge-based techniques. MRCS also provides a natural framework within which to combine different solutions, as in the more successful RTE solutions. The next two sections define MRCS's representation and comparison/inference architecture.

3 Modular Representation Scheme

MRCS's layered representation allows integration and comparison of arbitrary natural language text analysis from a variety of sources in a straightforward, uniform way, by representing each analysis source as a view of the underlying text span, populated with constituents appropriate to that source. Constituents may have attributes, and may be connected to other constituents by relations (including constituents in other views).

Figure 1 shows an example of this representation. The Named Entity view contains constituents corresponding to the named entities in the text span; a Syntactic Parse view, not shown in the figure, would contain a tree structure comprising Syntactic Parse constituents linked by ChildOf relations. These views are aligned with the underlying text (and therefore with each other). A Logical Form view could be added in a similar way.

The goal of MRCS is to represent such analysis as can be reliably induced from automated resources for a given span of text, in a way that is amenable to storage and reuse in NLP applications, and in a way that is easily extended to accommodate new NLP components or systems. It is also designed to support the composition of a range of local NLP functionalities such as context-sensitive Verb Paraphrasing and Named Entity resolution in a highly modular, configurable inference algorithm.

MRCS's underlying data structure is a directed graph whose nodes represent constituents of a text span, each node having an arbitrary set of attribute/value pairs, and whose edges represent relations between these constituents. A constituent could correspond to a word, a phrase, an entity, a predicate, a sentence, a paragraph, etc. Relations are typed connections between constituents, and could represent sequence, containment, or roles in a semantic frame. Constituents are organized into views, each of which corresponds to a separate type of analysis over the underlying text span. The views relating to a given text span are collected in a TextAnnotation. The fundamental Word view is present in all TextAnnotations, and records the tokens in the underlying text span from which all other analyses to be integrated are derived.

MRCS imposes two constraints on the structures it represents: 1) Each constituent and view must have a unique identifier; 2) Two constituents may be connected by many different relations, but only a single relation of a given type can connect two constituents. Although it is not a constraint per se, for simplicity we assume here that analysis structures are directed acyclic graphs.

In the remainder of this section, we define the element of MRCS' representation, and in the next, describe the comparison/inference architecture.

3.1 Definitions of MRCS Components

The MRCS Representation schema consists of the following components (where types are simply labels):

An **attribute** is a pair $\{T, M\}$ where T specifies the type and M the value of that attribute.

A **constituent** is a tuple $\{ID, T, \{A\}, \{R\}, \{N\}\}$, where ID is a unique identifier, T is a type, $\{A\}$ is a set of attributes, $\{R\}$ is a collection of relations to other constituents (all of which have ID as their source constituents) and $\{N\}$ is the set of underlying Word indexes to which the constituent corresponds.

A **relation** is a tuple $\{T, P, Q\}$, where T specifies the relation type, and P and Q are the Identifiers of its source and sink constituents. Examples of relation types are $Before/After$, $Contains$ and $Parent/Child$. The constituents a relation connects may be in different views.

A **view** is a collection of constituents. Views may correspond to individual sources of analysis, or be arbitrarily defined for convenience in a given application.

A **textAnnotation** is a collection of views.

The **underlying text span** is the sequence of words to be represented by a TextAnnotation.

In Figure 1, the Text and Hypothesis are each represented as a TextAnnotation. The underlying text span of the Text is the two sentences: "Italian Director Federico Fellini had many successful films. He eventually received an Oscar." Each supports a set of relevant views – the fundamental Word view, and views for Named Entities (NE), Coreference (Coref), and Semantic Role Labeling (SRL).

Semantic Role information (the SRL view), represents verbs as predicate constituents, while argument constituents represent the participants and adjuncts. Attributes describe the relevant properties of each constituent, such as the tense of the predicate's verb. Relations connect the arguments to the predicates, and their types indicate the roles of the arguments with respect to the predicate they connect to; in the SRL view, the ARG_1 relation indicates that its target constituent fills the ARG_1 (or "patient") role of the predicate that is its source.

3.2 MRCS Text-level Description Language

The MRCS Text-level Description Language (TDL) is a simple, extendable text form used to describe the TextAnnotation representation described above.

To represent grounded TextAnnotations like those in Figure 1, TDL uses symbols, predicates and the conjunction operator. Each symbol refers either to an Identifier, a Type or a Value.

A description of a TextAnnotation is a conjunction of terms describing attributes, relations and views.

$attribute(ID, T, V)$ specifies that the constituent with the unique identifier ID has an attribute with type T and value V. Constituent types and indexes are specified using attribute terms.

$relation(T, ID_1, ID_2)$ specifies that a relation of type T connects the constituent with the unique identifier ID_1 to the constituent with the unique identifier ID_2.

$view(ID_V, ID_C)$ specifies that the constituent with the unique identifier ID_C belongs to the view with identifier ID_V. (Note that the constraint that only one view of each type can exist in a TextAnnotation means that the view's type can also be used as its Identifier.)

Note that by parameterizing the predicates to take view and attribute type arguments, this logical form makes it easy to extend the language for new resources: one can simply specify new types, without extending the implementation of the language interpreter.

Consider the SRL view in the Text of Figure 1.

The constituent corresponding to the predicate "receive" has the following TDL representation:

```
viewMember(SRL, TV2)
attribute(TV2, VB, receive)
attribute(TV2, Tense, past)
attribute(TV2, INDEXES, 12)
relation(ARG_0, TV2, TV2A0)
...
```

4 Modular Inference Algorithm

The inference mechanism MRCS currently supports is a modular, configurable, unification-like architecture. This inference procedure depends on a set of decisions, each of which represents the comparison of (some element of) one text span, A, to (some element of) another, B, with the comparison determining whether A entails B. (Equivalence is modeled by determining also whether B entails A.) Local decisions are combined in a hierarchical fashion. In the context of RTE, a Hypothesis is considered to be entailed by a Text if it can be unified with it using some specified composition of local unification results.

In this section we outline the inference architecture, showing how behavior may be specified through the use of specialized comparators and combinators. We illustrate the definitions using the Textual Entailment example in Figure 1.

4.1 Inference Algorithm

MRCS's inference procedure is similar in concept to unification, but instead of exact subsumption, it uses specialized modules (*comparators*) to compare elements of one text span's representation with those in another. This compositional assumption allows the user to leverage local NLP solutions such as named entity mention resolution and context-sensitive verb paraphrasing in a clean, modular way. Using the alignment of constituents with the words underlying the text span, the algorithm allows for comparison of different types of constituent, even when not explicitly linked in some view.

The system architect specifies a *policy* that defines: 1) an iteration scheme over the views and constituents of the TextAnnotations, which specifies which views are core and which are auxiliary (see below); 2) a mapping from constituent types and attribute types to Comparator types; and 3) a set of conditions that must be satisfied for the Text to entail the Hypothesis.

The inference process iterates over views specified in the policy, comparing constituents from the Text and Hypothesis. The comparator(s) used for each comparison are determined by the mapping specified in the policy. The inference algorithm recursively compares neighbors of those constituents, seeking a valid entailment for each constituent in the Hypothesis by a constituent in the Text that respects the connectivity of entailed neighbors. Each comparison, which results in a real-valued number output, comprises a set of entailment decisions over the attributes of the two constituents, and over the entailment decisions for their neighbors. To compose these results, *combinators* are used, which represent some function mapping a set of inputs to a single real-valued output (for example, computing a mean, maximum, or minimum score). At present, combinator inputs and outputs are confined to real values in the range $[0, 1]$. However, they could be extended to use a feature representation of neighbor decisions in order to allow a classification approach to decision making.

Attribute comparators specify a set A of attribute types and encode a function $f(A(c_h), A(c_t)) \rightarrow \Re^+$. For example, an attribute comparator for words would compare attributes representing the surface form of lexical tokens of two constituents and use resources such as WordNet to determine whether one entails another.

Constituent comparators serve four functions when comparing two constituents, c_h and c_t: 1. using a set of attribute comparators appropriate to the attributes in the two constituents, compose the results of their decisions using an appropriate combinator; 2. use this result to decide whether or not to proceed with comparing neighbors; 3. determine pairings of neighbors of c_h with neighbors of c_t, based on constraints over edge types, and return this mapping to the inference algorithm; 4. when prompted by the inference algorithm, compose decision results for neighbors with the decision result for the attributes.

The constraints used to determine valid edge mappings are specified as part of the configuration of a given constituent comparator. The default constraint enforces equivalence of edge types (e.g., a ChildOf edge may be matched only by a ChildOf edge). This can be overridden by, for example, a constituent comparator for SRL predicates, which could use a list of mappings between verb pairs such as "A award B to C" to "C receive B from A" to determine constraints that link edges representing the $buyer$ and $sold - to$ roles for these predicates.

The inference algorithm is sketched below:

Algorithm 1 Inference Algorithm (sketch)

COMPARE-TEXT-ANNOTATIONS($Text, Hyp$)
$\{$Select C_h, C_t according to the specified policy$\}$
for all Constituents C_h in Hyp **do**
  for all Constituents C_t in $Text$ **do**
    COMPARE-CONSTITUENTS(C_h, C_t)
  end for
  Combine comparison scores for C_h
end for
Combine comparison scores for Hyp and return result
END COMPARE-TEXT-ANNOTATIONS

COMPARE-CONSTITUENTS(C_h, C_t):
Using the appropriate attribute comparators, compare the attributes of C_h with those in C_t
Compose the attribute comparison scores
if attribute score is sufficient **then**
  get neighbors $\{N_h\}$ of C_h
  for all $C_h\prime \epsilon \{N_h\}$ **do**
    find viable neighbors $\{C_t\}\prime$ of C_t
    for all $C_t\prime \epsilon \{C_t\}\prime$ **do**
      COMPARE-CONSTITUENTS($C_h\prime$, $C_t\prime$)
    end for
  end for
  Combine Neighborhood and Attribute scores
end if
RETURN combined score
END COMPARE-CONSTITUENTS(C_h, C_t)

The acyclicity requirement on the TextAnnotation structure guarantees that this process will terminate, as at the leaves of the graph, empty neighborhoods will be returned, and the recursive comparison step will be completed.

In the TE example in Figure 1, we specify a policy that compares SRL constituents in the Hypothesis to SRL constituents in the Text. We assume that since the Text must entail the Hypothesis, the Hypothesis is more general than the Text, and so for the overall match to succeed, we require that every constituent in the SRL view of the Hypothesis, C^s_{srl} be entailed by some constituent C^t_{srl} in the SRL view of the Text. We specify this behavior using a combinator that returns the geometric mean of its inputs.

The inference algorithm first compares C_{HV1} to C_{TV1} and C_{TV2}. This comparison will fail because the verb attributes in the Predicate constituents don't match. The algorithm next compares C_{HV2} to C_{TV1} and fails as before. However, the comparison with C_{TV2} will succeed: the attributes in C_{HV2} and C_{TV2} match. The neighbors C_{HV1A1} and C_{TV2A1} match. The algorithm compares C_{HV1A2} to C_{TV2A0} and this match fails. However, the algorithm finds the auxiliary coreference constituent C_{Co2} (see below), which matches C_{HV1A2}. All the neighbors of C_{HV1} have now been matched successfully, so the overall match succeeds. The algorithm returns a positive score, indicating that the Text entails the Hypothesis.

4.2 Auxiliary Views

If two constituents map to the same underlying words, they are assumed to be in some sense equivalent, alternative representations of those words, and therefore interchangeable. We therefore specify an *auxiliary* view as containing constituents representing supplemental information that *may* be matched in place of some other, corresponding constituent structure. In the TE example, we consider the SRL view to be the core view of each TextAnnotation, while the Named Entity and Coreference views are auxiliary.

The mapping of each view to $\{Core, Auxiliary\}$ is part of the policy specified by the system architect.

In Algorithm 1, instead of simply comparing two constituents, MRCS can find substitutes for each in auxiliary views in their respective TextAnnotations. In the TE example, when the algorithm compares the Argument constituent C_{TV2A0} from the Text with C_{HV1A2} in the Hypothesis, the match will fail as the word "he" does not itself entail "An Italian director". Using the auxiliary view mechanism, the algorithm can retrieve the Text coref constituent C_{TCo2}, substitute it for C_{TV2A0}, and the match succeeds.

This mechanism makes it straightforward to incorporate local information provided by a new NLP resource.

4.3 Leveraging Compare-Time Contextual Evidence

By augmenting the inference algorithm above with an auxiliary analysis step, the auxiliary mechanism described above can be used to infer missing analysis, such as named entity information. When an argument from a $Text$ SRL predicate is compared with one from the $Hypothesis$, and the

Hypothesis argument has an auxiliary named entity constituent, but the *Text* argument does not, the inference algorithm can add that information to the *Text* on the fly. This resolves one of the problems raised in section 2.

Additionally, if a comparator encapsulating the context sensitive verb paraphraser referred to above determines a valid replacement for a (possibly discontinuous) verb phrase in the *Text* using candidate verbs from the *Hypothesis*, the inference algorithm may use appropriate resources to generate a new version of the *Text* sentence, and re-analyze it, since an error of this kind is likely to distort multiple levels of analysis, including Semantic Role Labelling.

5 Solving Problems with MRCS

We now briefly address the ways in which MRCS can address the concerns raised in section 1.

5.1 Leveraging Comparison-Time Context

Given a component like the Context-Sensitive Verb Paraphraser (Connor and Roth, 2007), MRCS can reserve some annotation steps for Comparison Time, at which point it can use candidate verbs from the Hypothesis as candidate replacements for verbs and verb-phrases in the Text. If such replacements are found, the inference algorithm may allow the Text to be re-analyzed; this is particularly useful for non-contiguous verb phrases, which may result in poor analysis by statistical parsers and semantic role labelers.

5.2 Building a Hybrid Entailment System

Given a system like that of (Tatu and Moldovan, 2007), the representation used by each component can be encapsulated within a separate view. The policy for the inference algorithm will compose results for a shallow lexical approach using an appropriate view that is simply a collection of word-level constituents, will specify a comparator that accesses the appropriate lexical resources, and specifies an appropriate combinator function to combine these scores and compare them to a threshold. For a theorem-proving component, the appropriate view can contain a graph of logical terms, with a comparator that calls the theorem-prover with the Text and Hypothesis representations, and which outputs the confidence score. A combinator that combines these (and possibly other) components can specify a weighted expert model over its inputs, or encapsulate a classifier that uses shallow features extracted from the different comparator results or view representations to determine the final entailment decision.

6 Related Work

Two main architectures have been proposed for general NLP applications, GATE ((Cunningham et al., 2002)) and UIMA [1], which depend on similar representations. These appear to restrict the user to annotating contiguous text spans, which is problematic when representing (for example) Semantic Role structures with long-range dependencies. Moreover, both are directed towards pipelined automated analysis of input text, not towards the comparison of analyzed text structures.

In the PASCAL RTE Challenges (Dagan et al., 2006), the most successful systems (such as (Tatu and Moldovan, 2007)) composed a range of NLP resources and techniques. Systems that do not make use of a reasoning component appear to have strong limitations on their optimal performance, with the best shallow system (Adams et al., 2007) falling significantly short of the best hybrid systems. Approaches that focus solely on a KRR approach ((Bayer et al., 2005) and (Bobrow et al., 2007)) fare even worse, with low overall performance due to KB coverage problems and brittleness with respect to mistakes in preprocessing the text.

Recently, there have also been efforts to compose "off-the-shelf" components to build a system that can interpret natural language text and derive correct logical forms (Barker et al., 2007), with the ultimate goal of augmenting ontologies or other knowledge resources, but with limited success.

We have found no work presenting architectures suitable for investigating the trade-off between deep and shallow NLP techniques, flexibly incorporating arbitrary NLP components, or evaluating the contributions of individual NLP components. MRCS is an appropriate paradigm with which to address these needs.

7 Conclusions

The most successful RTE and QA systems combine multiple NLP resources, but often in an ad-

[1] domino.research.ibm.com/comm/research_projects.nsf/pages/uima.index.html

hoc way, and typically rely on analysis that does not take advantage of contextual clues available at run-time. There is a need to study knowledge representations that can aggregate information at multiple levels and facilitate reasoning with these different levels in a way that defers disambiguation decisions that may benefit from such context.

We have explained a representation and inference paradigm, MRCS (Modular Representation and Comparison Scheme), showing how it supports arbitrary overlaid annotations in a clean, modular fashion. Extending MRCS to represent confidence scores for individual layers of analysis is straightforward. We have demonstrated ways in which this architecture addresses disambiguation decisions in the context of Recognizing Textual Entailment, a problem that can be used to model many other major NLP problems.

Work is in progress to build an RTE system using MRCS, and to extend the representation and inference to accommodate feature extraction and rewrite rules over arbitrary patterns in the representation.

References

Adams, Rod, Gabriel Nicolae, Cristina Nicolae, and Sanda Harabagiu. 2007. Textual entailment through extended lexical overlap and lexico-semantic matching. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 119–124, Prague, June. Association for Computational Linguistics.

Barker, K., B. Agashe, S. Chaw, J. Fan, N. Friedland, M. Glass, J. Hobbs, E.H. Hovy, D. Israel, D. Kim, R. Mulkar-Mehta, S. Patwardhan, B. Porter, D. Tecuci, and P. Yeh. 2007. Learning by reading: A prototype system, performance baseline, and lessons learned. In *AAAI*, Vancouver, Canada.

Bayer, Samuel, John Burger, Lisa Ferro, John Henderson, and Alexander Yeh. 2005. Mitre's submissions to the eu pascal rte challenge. In *PASCAL Challenges Workshop on Recognising Textual Entailment*.

Blackburn, P., J. Bos, M. Kohlhase, and H. de Nivelle. 1998. Automated theorem proving for natural language understanding.

Bobrow, Daniel, Dick Crouch, Tracy Halloway King, Cleo Condoravdi, Lauri Karttunen, Rowan Nairn, Valeria de Paiva, and Annie Zaenen. 2007. Precision-focused textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 16–21, Prague, June. Association for Computational Linguistics.

Connor, M. and D. Roth. 2007. Context sensitive paraphrasing with a single unsupervised classifier. In *Proc. of the European Conference on Machine Learning (ECML)*.

Crouch, Richard. 2005. Packed rewriting for mapping semantics to KR. In *International Workshop on Computational Semantics*, Tilburg, The Netherlands.

Cunningham, Hamish, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. Gate: A framework and graphical development environment for robust nlp tools and applications. In *ACL*.

Dagan, I., O. Glickman, and B. Magnini, editors. 2006. *The PASCAL Recognising Textual Entailment Challenge.*, volume 3944. Springer-Verlag, Berlin.

Dang, Hoa Trang, Jimmy Lin, and Diane Kelly. 2007. Overview of the trec 2007 question answering track. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007) Question Answering Track*, Gaithersburg, Maryland, November.

Hobbs, J. R., M. Stickel, P. Martin, and D. Edwards. 1988. Interpretation as abduction. In *Proc. of the 26th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 95–103.

Moore, R. C. 1986. Problems in logical form. In Grosz, B. J., K. Sparck Jones, and B. L. Webber, editors, *Natural Language Processing*. Kaufmann, Los Altos, CA.

Pinkal, M. 1999. On semantic underspecification.

Schubert, L. K. 1986. From english to logic: Contexfree computation of 'conventional' logical translations. In Grosz, B. J., K. Sparck Jones, and B. L. Webber, editors, *Natural Language Processing*. Kaufmann, Los Altos, CA.

Tatu, Marta and Dan Moldovan. 2007. Cogex at RTE 3. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 22–27, Prague, June. Association for Computational Linguistics.