Recognizing Textual Entailment

Chapter 1

Recognizing Textual Entailment

1.1 Introduction

Since 2005, researchers have worked on a broad task called *Recognizing Textual Entailment* (RTE), which is designed to focus efforts on general textual inference capabilities, but without constraining participants to use a specific representation or reasoning approach. There have been promising developments in this sub-field of Natural Language Processing (NLP), with systems showing steady improvement, and investigations of a range of approaches to the problem. A number of researchers appear to have converged on some defining characteristics of the problem, and on characteristics of practical approaches to solving it. RTE solutions have been shown to be of practical use in other NLP applications, and other grand Natural Language Understanding (NLU) challenges, such as *Learning by Reading* [25] and *Machine Reading* [41] have emerged that will require similar problems to be solved. It is an exciting time to be working in this area.

Textual Inference is a key capability for improving performance in a wide range of NLP tasks, particularly those which can benefit from integrating background knowledge. Performance of Question-Answering systems, which can be thought of as potentially the next generation of search engines, is limited, especially outside the class of factoid questions; and the task of extracting facts of interest (such as "People who have worked for Company X") from a collection of plain text documents (such as newspaper articles) may require significant abstraction, synthesis, and application of world knowledge on the part of a human reader – and therefore of software required to perform the same task.

In this chapter, we specify a framework within which you can design and build an RTE system. First, we define the problem of Recognizing Textual Entailment and outline its applications to other tasks in Natural Language Processing. We then

define a framework for an RTE system, and show how it accommodates techniques used by successful RTE systems, describing key research in the RTE field (with a focus on system development), and showing how each system relates to the framework we have defined. We finish by addressing the pressing challenges in RTE research, and point the reader to useful resources.

We assume that readers are already familiar with the fundamental ideas of Machine Learning (ML), and its methodology of training/development/testing; our focus is on the practical difficulties of developing an application for RTE.

We have provided simple algorithms for all the key steps of the RTE framework. While they are deliberately simplified, and as a result not particularly efficient, these are sufficient to allow you to build a basic RTE system that is designed to allow expansion along multiple dimensions. In section 1.4, where we discuss key research investigating different approaches to RTE, we map each line of research to our framework at a high level (for full implementation details, the reader is referred to the original work, as these are beyond the scope of this chapter). This mapping will allow you to develop the relevant aspects of the system to pursue those approaches that interest you most.

1.2 The Recognizing Textual Entailment Task

In this section, we define the task of Recognizing Textual Entailment, explain the pros and cons of this formulation, and show why the problem is non-trivial. We show how RTE can be applied to a range of NLP tasks, and present some concrete examples of such applications.

1.2.1 Problem Definition

The task of Recognizing Textual Entailment (RTE) in the form we address in this chapter is defined by Dagan, et al. [13] thus:

Definition 1. *Textual entailment is defined as a directional relationship between pairs of text expressions, denoted by T - the entailing "Text", and H - the entailed "Hypothesis". We say that T entails H if the meaning of H can be inferred from the meaning of T, as would typically be interpreted by people.*

As noted in Dagan, et al. [13], this somewhat informal definition is based on (and assumes) common human understanding of language as well as common background knowledge.

An **entailment pair** is composed of a Text T and a Hypothesis H; usually, H is a short statement, and T is a longer span of text. Figure 1.1 shows a sample

Text: The purchase of Houston-based LexCorp by BMI for $2Bn prompted widespread sell-offs by traders as they sought to minimize exposure. LexCorp had been an employee-owned concern since 2008.

Hyp 1**:** BMI acquired an American company.
Hyp 2**:** BMI bought employee-owned LexCorp for $3.4Bn.
Hyp 3**:** BMI is an employee-owned concern.

Figure 1.1. Some representative RTE examples.

Text and three Hypotheses. The label of each Entailment pair is determined by multiple human annotators; the background knowledge required is not specified, and remains a latent factor in the labeling process. Often, when such knowledge is required, it is "static" – such as cause-effect relations, or locations of well known cities or landmarks (which do not change over time) – rather than facts like the name of the current president of the United States (which changes over time).

The specification of the RTE task also requires that the Text be an inherent part of the reasoning for inferring the truth of the Hypothesis: while background knowledge may *augment* that represented by the Text, it may not *replace* it. If, for example, an RTE system uses facts extracted from Wikipedia, it might have a statement that ascertains the nationality of a popular film star, which could be equivalent to a Hypothesis statement. However, if evidence for this fact is not present in the Text, the entailment label is "Not Entailed", even though the Hypothesis alone states a "true" fact.

The two-way RTE task requires that systems label each entailment pair as either *Entailed* or *Not Entailed* – i.e. either T entails H, or T does not entail H. In figure 1.1, the Text entails Hyp 1, but not Hyp 2, or Hyp 3.

The three-way RTE task introduces the concept of contradiction. We define contradiction in entailment based on de Marneffe, et al. [16]:

Definition 2. *The Hypothesis H of an entailment pair contradicts the Text T if a human reader would say that the relations/events described by H are highly unlikely to be true given the relations/events described by T.*

The three-way RTE task requires that systems label each entailment pair as either *Entailed*, *Contradicted*, or *Unknown* – i.e. either T entails H, or H contradicts T, or it is unknown whether H is true given T. In figure 1.1, the text T entails Hyp 1; Hyp 2 contradicts T; and the truth value of Hyp 3 is unknown given the information in T.

The difficulty of the task depends on the entailment pairs selected, and designing a suitable corpus is non-trivial. The corpora produced by PASCAL [1] and NIST [2] are challenging. All corpora except RTE 4 have separate development and test components, each having between 600 and 800 entailment pairs; RTE 4 has a single component of 1000 pairs. All these corpora are balanced, with approximately 50% having *Entailed* and 50% *Not Entailed* labels. In RTE 4 and RTE 5, the *Not Entailed* examples were further divided into two categories: *Unknown* and *Contradicted* (35% and 15% of total examples respectively).

Each corpus defines a set of 3 to 7 "tasks" that further divide the data; each task corresponds to the domain from which its examples were drawn (examples: "QA" for Question Answering, "IE" for Information Extraction; see the publications describing each challenge for more detail, e.g. Bentivogli, et al. [2]). Performance of systems varies across tasks, indicating significant qualitative differences between the examples in each; but since the task label is not available to deployed RTE applications, we will not take it into consideration here. (If task information is present, it is trivial to extend the implementation of the framework described here to take advantage of it, either by introducing a feature representing the task, or by using separately tuned/trained inference components for each task.)

In addition, some pilot tasks were introduced in RTE 3 (explanation, contradiction) and RTE 5 (search). The contradiction task was made part of the main task for RTE 4 and RTE 5. RTE 5 also introduced a Search pilot task, which we will not pursue further here; the interested reader is referred to the relevant publication [2].

A system that performs well on these corpora could be said to have achieved a good "understanding" of natural language text [3]. State-of-the-art systems had accuracies of $\sim 74\%$ on the two-way task (*Entailed* vs. *Not Entailed*) and $\sim 68\%$ on the three-way task on the two most recent challenges (RTE 4 and RTE 5).

In the rest of this chapter, we identify the challenges involved in the RTE task, define a general framework to tackle it, and describe relevant research in RTE, showing how it fits into this framework.

1.2.2 The Challenge of RTE

It is informative to consider the different steps a human reader must go through to determine the entailment labels of the entailment pairs shown in figure 1.1.

To recognize that Hypothesis 1 is entailed by the text, a human reader must recognize that 1) "company" in the Hypothesis can match "LexCorp" and that

[1] `http://pascallin.ecs.soton.ac.uk/Challenges/RTE3/`

[2] `http://www.nist.gov/tac/2010/RTE/index.html`

[3] This assumption is based on the standard Machine Learning practice of evaluating the performance of a system on held-out data that was not used in training or development.

2) "based in Houston" implies "American". She must 3) also identify the nominalized relation "purchase", and 4) determine that "A purchased by B" implies "B acquires A".

To recognize that Hypothesis 2 contradicts the Text, similar steps are required, with the difference that the reader must integrate the information that LexCorp is employee-owned, and must then infer that because the stated purchase price is different in the Text and Hypothesis, but with high probability refers to the same transaction, Hypothesis 2 contradicts the Text.

Hypothesis 3 consists entirely of words from the text, but asserts a relation that cannot be discerned from the available evidence, and so its label is "Unknown": it is possible that BMI is employee-owned, but it may not be.

Some of these steps we identify above relate to other tasks defined by the NLP/Computational Linguistics community, such as Named Entity recognition (recognizing that LexCorp and BMI are companies), Co-reference (different mentions of LexCorp refer to the same underlying entity), and Semantic Role Labeling (BMI did the buying, not LexCorp). Others may not; the relevant tasks have not yet been well-developed in isolation, though they may related to recognized problem definitions. Perhaps hardest of all are textual inference steps that require us to apply our understanding of the world to identify cause-effect relations, entailment relations, and abstraction over multiple statements to a general principle.

While it is not required that a computerized solution to the RTE challenge follow such steps or emulate such capabilities, the limited success of approaches not informed by the human process has encouraged researchers to try a divide-and-conquer approach motivated by intuitions of the human process. Researchers have had some success isolating specific capabilities such as normalizing numerical quantities (dates, rates, proportions, counts), and have leveraged solutions to linguistically motivated problems like syntactic parsers, and shallow semantic analytical tools like Named Entity recognizers.

It could be argued that the examples in figure 1.1 might be resolved by simple lexical matching; but it should be evident that the Text can be made lexically very dissimilar to Hypothesis 1 while maintaining the Entailment relation, and that conversely, the lexical overlap between the Text and Hypothesis 2 can be made very high, while maintaining the Contradiction relation. This intuition is borne out by the results of the RTE challenges, which show that lexical similarity-based systems are outperformed by systems that use other, more structured analysis, as shown in section 1.2.3.

1.2.3 Evaluating Textual Entailment System Performance

Definition 1 has been used as the basis of six research challenges by PASCAL [13] and then NIST [2]; these corpora are available to the public (the first three without restrictions, the second three subject to a user agreement; see the websites noted earlier). Definition 2 motivated a pilot study in the third challenge, RTE 3; the corpora for the main task in both RTE 4 and RTE 5 incorporated contradiction, and so were labeled for both the two-way and three-way prediction tasks. [4]

These research challenges have generated a lot of interest, and significant progress on the RTE problem. We describe some informative examples in section 1.4; for now, we present a general sense of the *performance* of state-of-the-art systems.

Figure 1.2 graphs the results of the two-way entailment task for all five RTE challenges through 2009 (shown as lines) and includes the performance of a lexical baseline (due to Mehdad and Magnini [35]) on each data set, represented as large dots, one on each challenge's line. The results for each challenge are sorted from weakest to strongest, and the horizontal length of each graph indicates the relative number of participants (so, RTE 4 had the most participants so far).

It is difficult to compare results from different years, as each year's corpus is different (drawn from different domains, and/or according to different guidelines). RTE 4 and RTE 5 saw a significant increase in the average length of Text, to ~ 40 and ~ 100 words respectively, and are considered more challenging than entailment pairs with shorter texts. The lexical baseline, which uses a threshold based on the overlap between Hypothesis and Text words, indicates a fairly consistent baseline difficulty level, of between 55% and 58% for four of the five challenges so far. The result for RTE 3 (2007) is markedly higher, and all system entries appear correspondingly higher than in other years, suggesting an "easier" entailment corpus. In all cases, the baseline score is at or below the median score for each challenge.

The upper range of system performance has also been fairly consistent. The longer texts in RTE 4 and RTE 5 increase the difficulty of the task by introducing more irrelevant signals (additional words, phrases, and sentences that are often irrelevant to the entailment decision), increasing the processing burden on RTE systems, and broadening the scope for entailment examples that require the integration of information from multiple sentences.

Due to the non-comparability of the RTE data sets, it is hard to draw strong conclusions from the numbers themselves, other than to gauge the difficulty of the task based on the relatively strong performance of the lexical baseline, and to observe that some systems are significantly outperforming this baseline.

[4] At the time of writing, RTE 6 was underway.

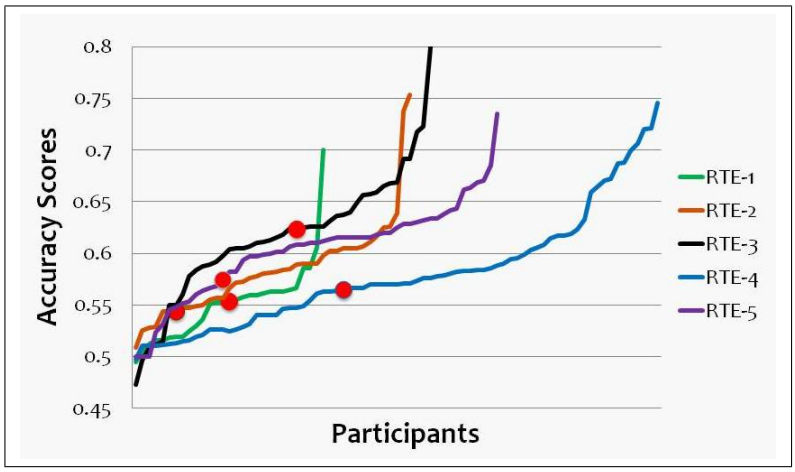

Figure 1.2. Results in Recognizing Textual Entailment Challenges 2-way task 2005-2009. The trace for each challenge sorts system results from lowest to highest accuracy; a longer trace indicates more participants. Red dots indicate performance of a lexical baseline system.

1.2.4 Applications of Textual Entailment Solutions

Many NLP problems can be formulated in terms of Recognizing Textual Entailment.

RTE clearly has relevance to Summarization [14], in which systems are required to generate human-readable summaries of one or more documents. The sub-task of identifying whether a new sentence contains information already expressed by a summary-in-progress (redundancy detection) can be thought of as an entailment pair with the present summary as the Text and the new sentence as Hypothesis. If T does not entail H, the sentence contains new information, and should be integrated with the summary.

Information Extraction is the task of recognizing instances of a fixed set of relations such as "works for" and "born in" in a set of natural language text documents. If we express the relations as short sentences, like "A person works for an organization", and "A person was born in a location", text spans from the source documents become the Texts of entailment pairs with the reformulated relations as Hypotheses, and an RTE system can be directly applied. Similarly, Question Answering, which requires automated systems to find candidate answers (sections of documents from a fixed document collection) to a set of questions, can be reformulated in much the same way: a question like "What is the largest city in South America?" can be reformulated as a short statement, "The largest city in South America is a city." This statement becomes a Hypothesis, and sections of the doc-

ument set – typically, paragraphs – become the Texts of a set of entailment pairs with this Hypothesis. An RTE system can be directly applied to identify actual answers.

Of course, these naïve reformulations of the IE and QA tasks are not in themselves sufficient, as RTE solutions are generally resource-intensive. However, the intuition is practical, as research that applies RTE to other Natural Language Processing tasks shows.

1.2.4.1 Question Answering

Harabagiu and Hickl [20] directly apply an RTE solution to re-rank candidate answers in a Question Answering system. The underlying idea is simple: a pre-existing Question Answering system returns the best candidate answers. While the top candidate may not be the correct answer, in many cases the correct answer is in the set of returned candidates.

Harabagiu and Hickl use an RTE system to assess each candidate answer. Their system first applies a rule-based implementation to transform the input question into a short statement, as illustrated above. A set of entailment pairs is created by combining each candidate answer in the set returned by the system as a Text, with the transformed question as the Hypothesis. The RTE system is then applied to each pair in turn: those candidates that entail the transformed question are moved to the top of the list, and those that did not are moved to the bottom. The study shows that including the Textual Entailment component improves system accuracy from 30.6% to 42.7%.

Celikyilmaz, et al. [5] use an entailment-like component to extract a feature-based representation of candidate question-answer pairs, after transforming the query in a similar way to Harabagiu and Hickl. They use the real-valued feature vectors derived from the entailment comparison to compute similarity values between members of a large set of question-answer pairs. These values are used as edge weights linking nodes representing individual QA pairs in a graph. A (small) subset of the QA pairs have gold-standard labels; the labels of the remaining nodes are then inferred using a semi-supervised learning method.

1.2.4.2 Exhaustive Search for Relations

In many information foraging tasks, such as patent search, accident report mining, and detecting confidential information in documents that must be shared with partners lacking appropriate clearance, there is a need to find all text snippets relevant to a given concept. This involves finding all passages that talk about the concept

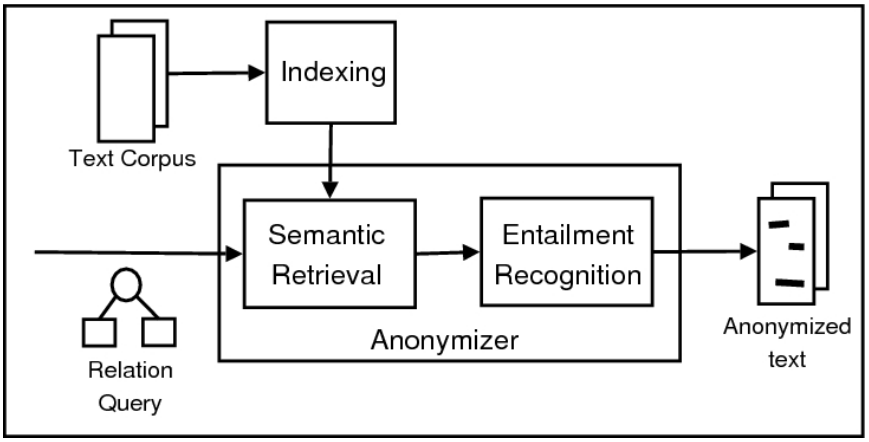

Figure 1.3. SERR framework [48]

directly or indirectly, while screening out passages that are superficially similar but have a different meaning.

This information need maps directly to recognizing entailed passages from large text corpora. However, this requires scaling up textual entailment systems to move from pairwise text-hypothesis decision to a search-based entailment framework. Since most successful RTE systems apply a lot of NLP resources and computationally expensive inference algorithms, a naïve approach (for every paragraph of each document, test whether it entails any one of a set of statements representing the target information) is impractical.

Roth, et al. [48] define a focused textual entailment approach, SERR (Scalable Entailment Relation Recognition), that consists of two stages: semantic retrieval and entailment recognition. Figure 1.3 shows the schematic diagram of the approach. The algorithm is outlined in figure 1.4. In this approach, the text corpus is first preprocessed to find semantic components such as named entities (people, location, organizations, numeric quantities, etc.). These are indexed as semantic units to facilitate quick retrieval. The user expresses the information need as a relation query, which is enriched with synonyms, alternate names, and other semantically similar keywords. This query is then used to retrieve text passages from the corpus. The results are processed by the textual entailment module to decide if the text entails the given query, and the entailed text snippets are then output as the results. The semantic retrieval helps improve the recall of entailing passages, while the RTE module filters the results to improve the overall precision.

The experimental evaluation was conducted using a corpus derived by taking all the Hypotheses from the IR and IE subtasks of RTE 1-3 as defining the infor-

SERR Algorithm
SETUP:
  Input: Text set D
  Output: Indices $\{I\}$ over D
  for each text $d \in D$
    Annotate d with local semantic content
  Build Search Indices $\{I\}$ over D

APPLICATION:
  Input: Information need S

  EXPANDED LEXICAL RETRIEVAL (ELR)(s):
    $R \leftarrow \emptyset$
    Expand s with semantically similar words
    Build search query q_s from s
    $R \leftarrow k$ top-ranked texts for q_s using indexes $\{I\}$
    return R

  SERR:
    Answer set $A \leftarrow \emptyset$
    for each query $s \in S$
      $R \leftarrow$ ELR(s)
      Answer set $A_s \leftarrow \emptyset$
      for each result $r \in R$
        Annotate s, r with NLP resources
        if r entails s
          $A_s \leftarrow A_s \cup r$
      $A \leftarrow A \cup \{A_s\}$
    return A

Figure 1.4. SERR algorithm, as described in [48].

mation needs, with all texts from the same entailment pairs forming a "document" set. The retrieval component found the most relevant documents (Texts) for each Hypothesis, and the RTE module labeled this returned set as "Entailed" or "Not Entailed" to identify the relevant documents.

When evaluated on the overall classification performance for the Hypothesis/Text pairs corresponding to actual examples from the RTE challenges, the system achieved performance in the top 3 ranks of published results for each challenge. The architecture also reduced the number of computationally intensive comparisons from $\sim 3,800,000$ for a naïve approach (compare all Hypotheses to all Texts using the RTE module) to only $\sim 40,000$ for the SERR system.

1.2.4.3 Machine Translation

Techniques developed by RTE researchers have also been applied to the task of evaluation in Machine Translation (MT). Padó, et al. [43] use insights from Textual Entailment to propose a new automated measure of candidate translation quality. MT evaluation uses statistical measures to evaluate the similarity of translations proposed by MT systems to *reference* translations produced by human annotators, as human evaluation of a large number of MT outputs is too resource-intensive to allow rapid evaluation of systems on large corpora. The dominant similarity metric is n-gram-based; while this measure has reasonable correlation with human judgments, it is far from perfect, not least because it takes no account of non-local structure in the translations to be compared.

Padó, et al. [43] propose a new metric that also accounts for *structural* characteristics, based on features similar to those used in the textual entailment system developed by Chambers, et al. [6]. Their intuition is that the candidate translation should be a paraphrase of the reference translation, and therefore the two translations should entail each other. Missing information in the candidate means it does not entail the reference, and additional information in the candidate means the reference does not entail the candidate. Bad translations will cause entailment to fail in both directions.

They use features based on the alignment score; modality, polarity, and tense mismatches; semantic relations; entity and date compatibility; and others. To evaluate their new metric, they use data from Machine Translation workshops. Their comparison shows a significant improvement in Spearman's correlation coefficient with human judgments over the standard metric.

Mirkin, et al. [40] use entailment to translate unknown terms. When a term is relatively rare, or when translating from a language with scarce linguistic resources, that term may not appear in the phrase tables used by MT systems. Mirkin, et al. tackle this problem by transforming the source translation into a more gen-

eral form by applying lexical entailment rules. They demonstrate the feasibility of this approach using a MT model trained on a parallel French/English corpus. They then apply this model to sentences from news articles in English, which have many unknown terms, being drawn from a different domain than that used to train the model.

Using English as the source language allows them to use WordNet [19], a large English language ontology relating words via synonymy, hypernymy, and many other lexical relations. They use synonymy to generate paraphrases for unknown words, and hypernymy to generate entailed (more general) texts from the English sentences. They then compare the quality of the French translations of these different versions of sentences with unknown words, to French translations using only the more standard paraphrase resources.

Their results show that the coverage of unknown terms over the paraphrase-based approach is improved by as much as 50% by the TE-based approach; translation quality is also much higher than when unknown words are omitted, with an additional 15.6% of translations produced by the system being judged "acceptable" by human judges, with only a 2.7% drop in the number of correct translations.

1.2.5 RTE in Other Languages

As yet, there are few entailment corpora in languages other than English. The two known sources of non-English RTE data are EVALITA [5] and the Cross-Language Evaluation Forum (CLEF) [6]. EVALITA, an Italian NLP evaluation program run by FBK-Irst of Trento, Italy, assesses NLP technologies for the Italian language on a range of problems that includes Recognizing Textual Entailment. CLEF's Answer Validation Exercise uses the RTE formalism to push Question Answering technology. CLEF develops corpora that pair candidate answers with questions reformulated as statements, with the idea that an RTE system can detect valid answers by determining whether each candidate answer entails the reformulated question; they have corpora for German, English, Spanish, French, Italian, Dutch, and Portuguese.

The NLP community has made steady progress in developing NLP resources comparable to those available for English in other languages: some good sources of information are the European Language Resources Association [7] and the Asian Federation of Natural Language Processing [8]. However, there are languages for

[5] http://evalita.fbk.eu/te.html
[6] http://nlp.uned.es/clef-qa/ave/
[7] http://www.elra.info/
[8] http://www.afnlp.org/

which resources such as Named Entity taggers and syntactic parsers have not yet been developed, requiring developers to use shallower cues for entailment.

One specific assumption we make in the framework we propose is that when a language has multiple resources, they are consistent in their determination of word boundaries. In reality, even English resources may be inconsistent in their tokenization of raw input text. Morphologically rich languages like Arabic may result in resources that segment individual words differently, by separating affixes and/or clitics. Languages like German that combine words to form unsegmented compounds also pose challenges to recognizing word boundaries. Chinese characters are not white space separated, and are grouped into word equivalents by NLP applications such as machine translators.

There is no one-size-fits-all solution; in the proposed framework, developers must determine the tokenization scheme that best suits their needs, and ensure that their different levels of representation respect the chosen tokenization; if different resources use conflicting tokenization schemes, it is the developer's task to satisfactorily resolve them. However, provided this requirement is satisfied, the framework we describe allows developers to implement a solution appropriate to the resources available to them.

1.3 A Framework for Recognizing Textual Entailment

In this section, we define a flexible framework as the basis of an RTE application; we draw on insights from Roth et al. [47]. To extensively define a single implementation of any real RTE system would easily fill a chapter by itself; instead, we describe a system, giving sample algorithms where appropriate. In the Case Studies (section 1.4) we describe some relevant research publications that provide details of specific implementations, and show how they fit into our framework.

The framework we specify here is designed to incorporate existing (and new) NLP resources in a uniform way, and to allow systematic development of both straightforward and complex RTE systems. It is also intended to directly support implementation of a wide range of approaches to RTE that have been described by researchers, such as those in section 1.4.

At the end of the chapter, we provide an incomplete list of resources that are available for download (most of which have a license for non-commercial use). However, here we avoid committing to any specific implementation; we focus on applications that perform well-established tasks, and therefore expect these applications to produce consistent output, so you should be able to use the specific applications that you feel suit your needs the best.

1.3.1 Requirements

Before designing a framework for a RTE system, it is instructive to consider the pre-existing NLP components that could be useful for determining entailment. We focus on applications that correspond to well-known NLP tasks that have a broadly-agreed output format, and which can clearly contribute to good performance on the RTE task, or which intuitively may form the basis of useful RTE functionality.

There are two main kinds of resource of immediate interest: resources that enrich raw text with semantic information, like Named Entity taggers and syntactic parsers; and resources that compare spans of text (such as individual words, or names, or phrases) and indicate some measure of similarity. We will refer to the former as *Annotators* (or, equivalently, *Analytics*), and the latter as *Comparators* or *Metrics* (see section 1.3.3.2).

Consider again the illustrative example in figure 1.1, and the steps a human reasoner must follow, as a guide to the capabilities our system must support.

In step 1, it is necessary to identify the entities BMI and LexCorp, and moreover, to recognize that they are Companies. This information, or something like it, can be provided by a Named Entity Recognizer. Step 2 involves mapping "based in Houston" to "American". One possible route to attain this connection is to infer "in America" from "in Houston". This requires a factoid knowledge base that operates at least at the lexical level. To recognize the nominalized relation "purchase" in step 3 first requires that this word be identified as a noun (provided by a Part-Of-Speech tagger). To map arbitrary nominalized verbs to their regular forms requires a lexicon; one possibility is the popular lexical ontology WordNet [38] (using the relation "derivationally related form" to identify the verb "purchase"). To make the next step requires interpretation of the syntactic structure to identify the subject, object, and direct object (the *arguments* of the nominalized verb), in order to allow comparison with the structure "BMI acquired an American Company". This structure could be obtained using a Syntactic or Dependency Parser; alternatively, these steps might be resolved by a shallow semantic parser (or "Semantic Role Labeler"). Finally, in step 4, the two syntactic (or shallow semantic) structures must be compared in order to recognize that the Text entails the Hypothesis.

There are other resources that could be useful; successful RTE systems also use resources that:

- identify and normalize numeric quantities
- identify different ways of expressing a given named entity (for example, "International Business Machines" might be referred to as "IBM", but not "BMI")
- determine which entities in a span of text refer to the same underlying entity

(aka. co-reference resolution)

Individual implementations generally use a mixture of off-the-shelf applications and custom-built modules, but attack the same set of underlying problems.

The implications for a general-purpose RTE framework are that it must support a range of annotations of the text of the entailment pair at multiple granularities (from words to phrases to verb-argument structures); and that it must support comparison of these annotations using specialized resources.

1.3.2 Analysis

The range of NLP resources described above presuppose that natural language understanding is (largely) compositional: we can isolate individual phenomena, and solve the task of recognizing each phenomenon. We see a similar intuition at work in the way human annotators describe solving entailment problems, as in section 1.2.2. Experience in a range of other fields of Computer Science give testimony to the power of the divide-and-conquer approach. We therefore seek a way to apply this strategy to the RTE problem.

1.3.3 Useful Components

We define here some components that are broadly useful in our generic RTE framework.

1.3.3.1 A Multi-view Representation for NLP Analysis

We consider the output of all analytics to define *constituents* over the underlying text, which may optionally be linked by *relations*. We refer to any pattern over constituents and/or relations as a *structure*. Each constituent is trivially a structure.

We consider each analytic resource to define its own *view* of the underlying text, with the most fundamental view being the *Word* view. We require that the Word view represent tokens, rather than the raw text, and that all other views be normalized with respect to this view – that is, the word tokens used to generate every view must be the same as those in the word view. As a consequence of this constraint, every constituent must correspond precisely to a set of word indexes. This is convenient when detecting correspondences across different views (for example, recognizing that an SRL argument is also a Named Entity).

Figure 1.5 illustrates the data structure generated from input of a system that combines Named Entities (NE), Numerical Quantities (NUM), and Semantic Role Label analysis (SRL), in addition to the words and their indexes. Each constituent

corresponds to words in the original text, and contains the list of indexes to which it corresponds.

In general, constituents specify: a type (used to select from available similarity metrics); one or more attribute-value pairs (specifying information of interest – such as Part-of-Speech and lemma for words – and which may be used by the relevant similarity metrics); and the set of indexes of words in the original text to which the constituent corresponds.

In the example shown, the Named Entity constituents have a type and a value, derived from the output of a Named Entity Recognizer. The Numerical Quantity constituents have a normalized representation of the number and its unit, together with the indexes of the corresponding tokens in the original text. The Semantic Role Labeling view contains *predicate* (P) and *argument* (A) constituents; these are joined by relations representing the roles of the arguments related to the predicate (A0 is "agent", or *semantic subject*; A1 is "patient", or *semantic object*). These roles are distinct from syntactic subject and object roles, as they are not affected by e.g. passive constructions (for a full explanation of Semantic Role Labeling, see Palmer, et al. [44]). Note that nesting is allowed in our representation – a predicate may take another predicate as its argument (in this case, *say* has the predicate *buy* as its semantic object). The argument constituents themselves are not assigned roles because they may be part of more than one predicate-argument structure, and could have different roles in each.

The Multi-View representation has several key advantages over simpler, unified representations. Each resource is handled independently of the others, and can be added incrementally. The representation is also very flexible: if you want to use different information sources for different purposes – as in the case of filtering (see section 1.3.7) – it is straightforward. It is also possible to write generic algorithms for processing multiple views without knowing what views will be present. Finally, the Multi-View representation defers *Canonization*: combining the different views into a single data structure may require resolving disagreements in boundaries and relation structure, and it may be desirable to make these decisions at a later stage – for example, during the inference step, when there may be additional evidence to support one decision over another. If desired, the many views can be collapsed into a single graph structure as the final step of the preprocessing stage.

1.3.3.2 Comparing Annotation Constituents

A crucial step in RTE is that of comparing the Hypothesis with the Text. Given our integration of many different information sources, we will need specialized resources to compare some types of constituents. We will simplify our implementation if we treat these resources in a uniform way, so we use the *Metric* abstraction:

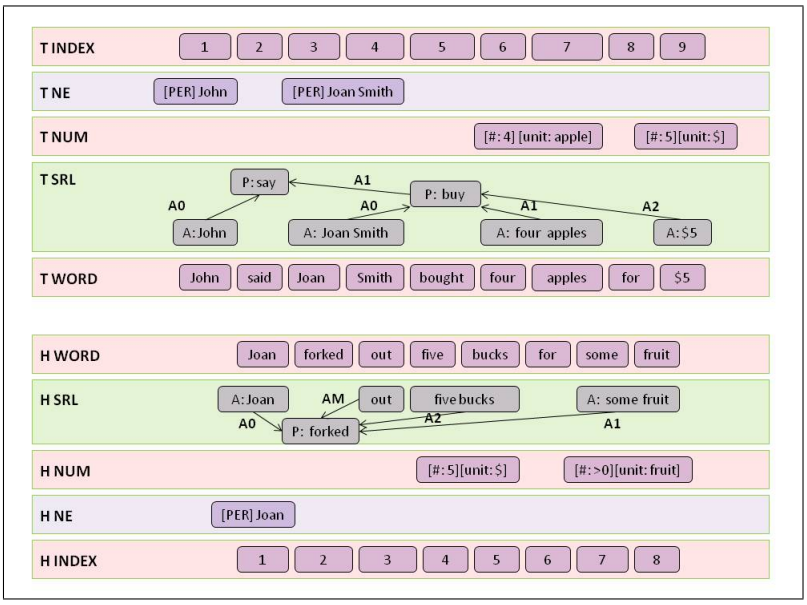

Figure 1.5. Example of a Multi-View representation of a Textual Entailment pair.

Definition 3. *A Metric compares two constituents and returns a real number in the interval [-1, 1], with the value* 1 *indicating identity,* -1 *indicating opposition, and* 0 *indicating irrelevance.*

The Metric is a specialization of the concept of *Comparator*. A Comparator compares two structures and returns arbitrary information; a Metric compares two constituents and returns only a score. Comparators tend to be more specialized, being designed to work with specific structures (such as Predicate-Argument structures derived from Semantic Role Label annotation).

Note that this definition of metric limits consideration of the context, except for the knowledge of the type of constituents being compared, and whatever information is encoded by the analytical resource that generates the input used to create the constituents, and by the algorithm that parses that input into the constituents. We think of metrics as fairly simple, focused resources, and chose the Metric abstraction to allow us to specify a simple interface and thereby simplify graph generation code.

One reason for this design choice becomes clear when you consider what happens when you add a new information source to an existing, possibly complex, RTE system: ideally, you want to avoid rewriting your graph generation and alignment algorithms. If the new comparators you write to handle the new annotation follow

the same specification as the others you have already written, you should not have to change these algorithms. Another reason for this localization is to promote the encapsulation of domain-specific knowledge in a convenient form.

To give a concrete example of a metric, we will describe the behavior of a word metric (the algorithm is given in figure 1.6).

Given the pair of word constituents with the lemmas "rise" and "increase", our word metric should return a high positive score such as 0.8, as these words are synonymous in some contexts. Given "paper" and "exterminate", it should return a value near 0 as these two words are generally unrelated. If called with "rise" and "fall", it should return a negative value close to -0.7, as these are antonymous. (We use a smaller magnitude negative score so that our alignment step will prefer positive matches over negative ones, but this decision is based on our intuition of desired behavior rather than empirical knowledge.)

The determination of scores is presently more of an art than a science; we leave them as real values to retain flexibility in inference. We have found, for example, that the lexical baseline we use in our experiments performs better when using an admittedly imperfect real-valued word similarity score, compared to the case where the word similarity metric is thresholded to assign either 1.0 or 0.0.

In the general case, some metric scores may need to be adjusted. For example, some Named Entity similarity metrics use variations on string edit distance; these tend to return moderate positive scores for very dissimilar names. A word similarity metric based on WordNet, however, might return a relatively low positive score for two words related by several steps of hypernymy; yet this would certainly be more likely to coincide with a case where an entailment relationship holds between the two words, than the two entities with a similar score via a string edit distance.

Note that generally, metrics are NOT symmetric, because the entailment relation is not symmetric. Consider the case of a metric that compares noun phrases being applied in the entailment example in figure 1.1. One of the phrase pairs that must be compared comprises "a company" from the Text, and "an American company" from the Hypothesis: in this case, the Text does not contain sufficient information, and so the noun phrase metric should return a score of zero. However, if "an American company" were from the Text, and "a company" from the Hypothesis, the metric should return a score close to 1.0, as the first entails the second.[9] A Named Entity metric should recognize that "John Q. Smith" entails "John Smith" and "Mr. J. Smith" with high likelihood, but not "Ms. J. Smith"; again, this relationship is not always symmetric, as "John Smith" does not necessarily entail "John Q. Smith".

[9] The effect of additional modifiers (in this case, "American") on entailment is called *monotonicity*; for a discussion of entailment and monotonicity, see MacCartney and Manning [31].

```
// Assume: words both set to lowercase

compare( firstWordC, secondWordC )
   score ← 0
   firstWord ← getAttribute( firstWordC, WORD )
   secondWord ← getAttribute( secondWordC, WORD )

   if ( firstWord == secondWord )
        score ← 1.0
   else
        levDistance ← levenshteinDistance( firstWord, secondWord )
        numChars ← max( firstWord.length, secondWord.length )

        if ( ( numChars − levDistance ) / numChars > 0.9 )
             score ← 0.8
        else if ( isSynonym( firstWord, secondWord ) )
             score ← 0.9
        else if ( isAntonym( firstWord, secondWord ) )
             score ← −0.7
        else
             numHypernymLinks ← isHypernym( firstWord, secondWord ) )
                  if ( numHypernymLinks < 4 )
                       score ← (0.9/numHypernymLinks)
   return score
```

Figure 1.6. Algorithm for a word metric. The function "levenshteinDistance()" computes the edit distance between two strings. The function "isSynonym()" consults WordNet and returns "true" if the two words are synonyms, "false" otherwise. "isHypernym()" consults WordNet and returns the number of Hypernym links separating the two words (infinity if there is no link).

1.3.4 A General Model

A block diagram of a typical RTE system is presented in Figure 1.7. Entailment pairs are processed either one at a time or as a batch; for simplicity, we will describe the process per pair except in specific contexts that require a batch-processing mode. We will describe the system in terms of its *evaluation* (which corresponds to the behavior of a deployed RTE system); we will handle the process of training machine-learning components separately, though this process usually uses many of the same steps.

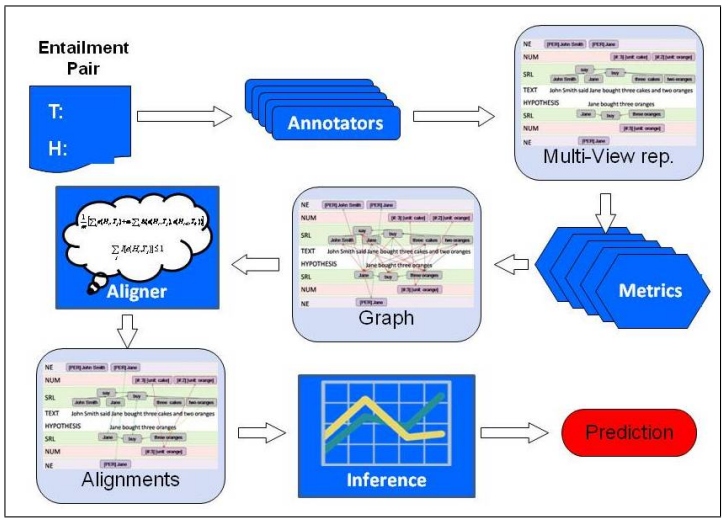

Figure 1.7. Block Diagram for Generic RTE Framework

1.3.4.1 Preprocessing

We assume that as the first step in the RTE process, our system must apply a suite of off-the-shelf [10] annotators to the text of the entailment pair. While the list of resources is open-ended, typical resources include: sentence and word segmentation (identify sentence boundaries, word and punctuation tokens); Part-of-Speech tagging; dependency parsing or syntactic parsing; Named Entity recognition; Co-reference resolution; Semantic Role Labeling. These different resources are used to enrich the text [11].

[10] We use the term "off-the-shelf" to describe packages or components that are readily available from one or more open-source/academic sources.

[11] The terminology of RTE is overloaded. We use **t**ext [span] to describe generic sentences, paragraphs, or portions of same, and **T**ext to refer to the larger component of an entailment pair.

We describe a data structure suited to integrating such diverse annotations in section 1.3.3.1, and where appropriate, show how this can be mapped to the types of representations used in some specific RTE systems.

Depending on the off-the-shelf components used, you may also need to clean up the input prior to applying these resources, but we know of no pre-packaged solutions. Some older packages may not handle multi-byte characters, for example: these must be replaced or omitted. A clean-up step could also normalize spelling, which may have a significant impact on e.g. syntactic parsers and POS taggers.

1.3.4.2 Enrichment

We use the term *enrichment*, as distinct from *preprocessing*, to refer to resources that operate on (combinations of) pre-existing views to either augment existing views, or to generate new views – in contrast to analytical resources, that process text and generate an annotated form that is directly parsed into constituents, relations and views. Enrichment resources serve one of two functions: to abstract over some text/annotation patterns by mapping them to a closed set of structures; or to augment the existing annotation by recognizing implicit content in the input text/annotation and making it explicit as new structure.

An example of abstraction would be to represent modifiers of verbs such as "failed to" in the sentence "Attackers failed to enter the building", or "said that" in the example shown in figure 1.5, by using an attribute in the verb or relation node in the corresponding predicate-argument structure. In the latter case, we could write code to identify such structures and mark embedded predicates like "buy" with an attribute indicating uncertainty.

An example of augmentation is that of rule application (see section 1.4.3), to make implicit content of the underlying text more explicit or to generate explicit paraphrases of the text. The RTE system may use them to generate additional syntactic parse trees representing paraphrases of the underlying text, or predicate-argument structures like those encoding Semantic Role Labeling information.

1.3.4.3 Graph Generation

After identifying various syntactic and semantic structures in the Text and Hypothesis, it is necessary to compare those in the Hypothesis with those in the Text. In the simplest systems, only words are compared. In more successful systems, a range of annotation types are compared. Typically, the Text and Hypothesis are represented as graphs whose nodes correspond to annotation units (such as words, Named Entities, parse sub-trees, SRL verb arguments), and whose edges correspond to connections within an annotation type (e.g. connecting different mentions

of a single entity via co-reference edges, or linking words in a dependency tree with typed dependency edges). Then, the constituents in the Hypothesis are linked to those in the Text based on some measure of similarity (possibly simple equality), to form a bipartite graph distinct from the Text and Hypothesis structures.

We assume that each type of constituent that must be compared in the graph generation step has an associated comparator (or *metric*, as defined in section 1.3.3.2). More than one type of constituent may use the same comparator, and comparators for complex constituents (those with structure, such as Numerical Quantities or Predicate Argument structures) may themselves call other, more basic comparators.

1.3.4.4 Alignment

The intuition behind the alignment stage, made explicit by de Marneffe, et al. [16], is that only a relatively small portion of the Text is relevant to the Hypothesis. The goal of alignment is to identify that relevant portion and thereby simplify the inference step.

Many RTE systems have an explicit alignment step; others have an integrated alignment/inference process. In general, alignments map each constituent in the Hypothesis to a single constituent in the Text. This is a heuristic based on the observation that the hypothesis tends to be much shorter than the text, and that in positive entailment examples, a human reader can often generate a "piecewise" explanation of the hypothesis using portions of the text.

Most RTE systems first integrate all constituents into a single graph structure – a single view in our terminology – and align each constituent in this representation. Others perform an alignment using only words, and in the inference step analyze the structure in other views that corresponds to the aligned words. In our own work [50] (described in section 1.4.6), our system performs multiple alignments for different groups of views, and the inference step compares them to discern cues for entailment/non-entailment.

1.3.4.5 Inference

All RTE systems must use a decision component to label each entailment pair. This may be a relatively simple measure of overlap plus a threshold, or it may be significantly more complex – for example, extracting features from the alignment graph and applying a machine-learned classifier to determine the final label. Some use theorem-provers over a logical representation induced from the entailment pair and the analysis from the preprocessing step. We discuss some different approaches in section 1.4.

Text: John said Joan Smith bought three apples for five dollars.

Hyp: Joan Smith forked out $5 for three apples.

Figure 1.8. A Textual Entailment pair for implementation examples.

1.3.5 Implementation

In this section, we fill out the explanation of the different parts of the RTE system sketched above, with a focus on functionalities common to a range of successful RTE systems. The case studies we consider in section 1.4 – drawn from recent RTE challenges – will be mapped onto these descriptions. For this general framework, we will use as our running example a simple Lexical Entailment Algorithm (LEA) that uses a WordNet-based similarity measure for Word constituents, and a simple Named-Entity-based filtering rule.

We will use as our sample input the (slightly contrived) entailment pair shown in figure 1.8. This example will allow us to illustrate each step of the RTE framework in the context of the LEA system.

1.3.5.1 Preprocessing

You will need to write the modules to control the flow of data through the various analytical resources, and to translate from the output of each resource to the constituent/relation/view data structures. Word-level annotation like Part-of-Speech and Lemmas can be integrated into Word constituents. Shallow annotation like Named Entities are straightforward to parse into constituents in their own view; structured annotation like Co-reference, and Semantic Role predicates and arguments, require some decisions about representation – for example, whether to have separate views for predicates and arguments, or whether to create additional constituents that each correspond to a complete SRL structure.

A typical order for preprocessing is: 1. Split into Sentences; 2. Split into words; 3. Part-Of-Speech (POS) tagging; 4. Dependency or syntactic parsing; 5. Named Entity Recognition; 6. Co-reference resolution (identify referents of pronouns and possibly other entity mentions); 7. Semantic Role Labeling (verbs and nominalized verbs). This ordering reflects some typical dependencies – for example, many NLP applications require POS tags as an information source, and most SRL systems require syntactic or dependency parse information. Some tools may allow or even expect the user to provide these inputs, while others handle everything internally. Providing such inputs yourself can improve efficiency by avoiding repeated application of tools with comparable functionality. For convenience,

word-level annotations like POS and Lemma can be added to Word constituents.

Note that if you use resources from different sources, they may have different expectations about input. For example, many applications take unsegmented text as input, and segment the text internally. The problem here is that there is no clear set of guidelines for "correct" segmentation, and so the output from different sources may disagree about some word and sentence boundaries. For example, should hyphenated words be separated (e.g. "American-led" vs. "American - led")? Should symbols representing currency remain with the corresponding number, or not (e.g. "$12M" vs. "$ 12 M" or "$ 12M")? In such cases you will need to resolve the differences yourself. Of course, you could use an integrated tool set that provides all the different kinds of annotation you need, or restrict yourself to tools that accept pre-segmented input. However, it is seldom the case that the tools with the highest performance in each task all come from the same source; and if a specific tool has been developed using a specific segmentation scheme, it may not perform as well when it is given input that uses a different segmentation scheme.

Running Example – Lexical Entailment Algorithm (LEA):

For our LEA RTE system, we will need two views: a Word view and a Named Entity view. The Word view will contain a Word constituent for each token in the corresponding entailment pair member (i.e., either the Text or the Hypothesis), which will include the original word and its lemma (if it has one). The Named Entity view will contain one constituent for each Named Entity in the corresponding entailment pair member, containing the entity's original representation (the sequence of tokens from the original text) and its type. We won't initially use all this information, but it will enable us to suggest possible extensions to the original algorithm. The resulting multi-view data structure is the same as that in figure 1.5, without the SRL and NUM views.

While some NLP applications provide a programmatic interface, many do not; however, almost all generate marked-up text output. For those unfamiliar with the task of parsing NLP tool outputs, we've outlined an algorithm to parse the NER output in figure 1.9, which also shows a sample NER output. We assume that either the NER segments the input text in the same way as that used to induce the Word view, or that the NER takes tokenized text as input. We also assume that there is no overlap of Named Entities in the NER output, an assumption that holds for the NER taggers we have used, though it is not hard to extend the algorithm to handle outputs of tools allowing tagged entities to overlap.

```
// sample nerOutput: "[PER Jane Smith ] bought apples ."

// ASSUME: no overlapping entities, and that square brackets
// in input have been replaced.

CreateViewFromNerOutput( String nerOutput )

    neView ← ∅
    neType ← null
    neValue ← null
    indexSet ← ∅
    isInNe ← false

    while ( nextWord ← getNextWord( nerOutput ) )

        firstChar ← peekNextChar( nextWord )
        if ( firstChar == '[' )
            isInNe ← true
            getFirstChar( nextWord )
            neType ← nextWord
        else if ( firstChar == ']' )
            neConstituent ← { neType, neValue, indexSet}
            neView ← neView ∪ neConstituent
            indexSet ← ∅
            neType ← null
            neValue ← null
            isInNe ← false
        else if ( isInNe )
            wordIndex ← wordIndex + 1
            indexSet ← indexSet ∪ index
            neValue ← concatenate( neValue, nextWord )
        else
            continue

    return neView
```

Figure 1.9. Algorithm to parse NER-style annotations. The function "getNextWord(nerOutput)" splits the first word from "nerOutput" at the first non-initial whitespace character and returns it; "peekNextChar(aWord)" returns the first character of "aWord"; and "concatenate(startString, nextWord)" appends "nextWord" to "startString" separated by a single whitespace character.

1.3.5.2 Enrichment

To extend our LEA system, we will enrich our underlying text by adding simpler expressions equivalent to idiomatic usage. This simplistic resource will use a hand-generated mapping from simple idiomatic phrases to simpler equivalent expressions, e.g. "kick the bucket" to "die". Provided we consider only those expressions that can be mapped to the same number or fewer of replacement words, we can simply add alternative word constituents that correspond to the same indexes as the original idiomatic expression (a single replacement word constituent may cover more than one of the original sentence indexes). A naïve algorithm for the IdiomMapper is shown in figure 1.10.

The enriched Multi-View data structure is presented in figure 1.13. The original Hypothesis text is "Mr. Smith forked out $5 for three oranges.", and the multi-view representation has a word constituent for each token, including the period. The IdiomMapper has added the new word constituent "pay". Note that this constituent covers both of the indexes that the original idiom "forked out" covered. This is important when determining optimal alignments (see section 1.3.6).

1.3.5.3 Graph Generation

In the graph generation step, the comparison resources (metrics) are applied to the relevant constituent pairs drawn from the Text and Hypothesis. This can be implemented in a straightforward way – iterate over views in the Hypothesis and Text, iterate over the constituents in each, and apply the appropriate metrics. The metric code may itself be complicated, however, for highly structured constituents like dependency parse (sub-)trees.

We provide a simple graph generation algorithm in figure 1.11.

Running Example:

In our example, we have the Named Entities "John" and "Joan Smith" in the Text and "Joan" in the Hypothesis. "John" and "Joan" have a very low edit distance (of 1), but a human reader knows that unless there is a typographical error, these two names refer to different people. We will assume that our Named Entity metric is smart enough to know this too, and that it will return a similarity score of -0.7.

The two strings "Joan Smith" and "Joan", our other Text-Hypothesis Named Entity pair, should return a high score, even though their edit distance (of 6) is relatively high. We assume our NER metric returns a score of 0.9, since the strings are not identical, but are highly likely to refer to the same individual.

We will assume that our Word similarity metric uses WordNet, and applies the

```
// ASSUME: annotationGraph already has word view;
// idiomList is a map from idiom strings to single words
// such as "forked out → buy"

AddIdiomView( annotationGraph )
  maxWordsInIdiom ← 3
  indexes ← getOrderedWordIndexes( annotationGraph )

  foreach index ( indexes )
      indexSet ← ∅
      offset ← 0
      sequence ← ""
      replacement ← null

      do
          offsetIndex ← index + offset
          word ← findWordWithIndex( annotationGraph, offsetIndex )
          sequence ← concatenate( sequence, word )
          replacement ← findIdiomMatch( sequence )
          indexSet ← indexSet ∪ offsetIndex
          offset ← offset + 1
      while ( ( replacement != null ) AND ( offset < maxWordsInIdiom ) );

      if ( replacement != null )
          idiomConstituent ←generateIdiomConstituent( replacement, indexSet )
          idiomView ← idiomView ∪ idiomConstituent

  if ( idiomView != ∅ )
      addView( annotationGraph, idiomView )

  return
```

Figure 1.10. Simple algorithm for generating Idiom view.

following heuristic: if words are linked by synonymy or one level of hypernymy, the score is 0.9. If they are linked by two levels of hypernymy, the score is 0.6. If they are linked by three levels of hypernymy, the score is 0.3. If the words are linked by antonymy, the score is -0.5. This behavior is specified in the algorithm shown in figure 1.6.

1.3.6 Alignment

The fundamental idea behind most alignment algorithms is the notion that some alignments are better than others, and that simply picking the most similar Text constituent for each Hypothesis constituent is too simplistic, as it does not account for sentence structure.

Given our formulation of comparators (metrics) and the method for generating the entailment graph, we can frame the task of finding an optimal alignment as an optimization problem. We will align *groups* of views together – for example, we could combine Named Entity and Numerical Quantity views in a single alignment. We may align all views together, simultaneously; or we may align each separately, depending on the type of inference we want to perform.

We constrain the alignment to allow each *index* in the hypothesis to be mapped to *at most one target* in the text, so constituents covering more than one token may not overlap. The goal is to identify parts of the Text that *explain* the tokens of the Hypothesis, and to simplify the inference problem.

In general, our intuition is that some views should *compete*: when there are several alternative representations of the same token(s) – such as substitutions for idioms – we may wish these to be considered as mutually exclusive choices, in which case these views should be grouped before alignment; we wish other views to be handled separately, because they may give us useful information that would be lost if they were grouped. For example: suppose a Named Entity metric returns only scores in the range $[0, 1]$, and no entity constituents match. If we combine the NE view with the Word view, we may get spurious matches of parts of entities that share a title, or a surname, or that have a regular noun as either a forename or surname that happens to appear in the other entailment pair member. A similar problem arises when we combine views using metrics that do not have compatible output (i.e., their scores cannot be interpreted in the same way). Again, combining Named Entities and Words may result in problems because the Word similarity metric consistently returns lower scores for positive matches. Constituents at different granularities may both have alignment edges in an optimal solution, provided they do not overlap.

Since metrics may return negative scores, the objective function must account for these. Negative scores indicate contradiction: in the absence of a better positive

```
CompareHypothesisToText( hypGraph, textGraph )
  edgeList ← ∅
  foreach view hypV in hypGraph
      viewEdgeList ← ∅
      foreach view textV in textGraph
          if ( isCompatible( hypV, textV ) )
              viewPairEdgeList ← CompareViews( hypV, textV )
              viewEdgeList ← viewEdgeList ∪ viewPairEdgeList
      edgeList ← edgeList ∪ viewEdgeList
  return edgeList


CompareViews( hypView, textView )
  edgeList ← ∅
  foreach constituent hypC in hypView
      hypEdgeList ← ∅
      hypId ← getIdentifier( hypC )
      foreach constituent textC in textView
          textId ← getIdentifier( textC )
          score ← CompareConstituents( hypC, textC )
          matchEdge ← { ViewType, hypId, textId, score }
          hypEdgelist ← hypEdgeList ∪ matchEdge
      edgeList ← edgeList ∪ hypEdgeList
  return edgeList


CompareConstituents( hypC, textC )
  hypType ← getType( hypC )
  textType ← getType( textC )
  comparatorSet ← getCompatibleComparator( hypType, textType )
  matchScore ← 0
  foreach ( comparator ∈ comparatorSet )
      score ← comparator → compare( hypC, textC )
      if ( score > matchScore )
          matchScore ← score
  return matchScore
```

Figure 1.11. Algorithm for the Graph Generation step (comparing entailment pair member graphs). It is assumed that the system stores a mapping from paired constituent types to compatible Comparators, and that Comparators behave like Metrics in returning a score.

match, this information may be highly relevant to the subsequent entailment decision. In the objective function, therefore, the *magnitude* of the edge weight is used. The edge retains a label indicating its negativity, which is used in the inference stage.

For alignments over shallow constituents, we must guess at the deep structure; we therefore include locality in the objective function by penalizing alignments where neighboring constituents in the hypothesis are paired with widely separated constituents in the text. We ignore crossing edges, as we do not believe these are reliably informative of entailment.

The objective function is then:

$$\frac{\sum_i e(H_i, T_j) + \alpha . \sum_i \Delta(e(H_i, T_j), e(H_{i+1}, T_k))}{m} \tag{1.1}$$

and the constraint:

$$\sum_j I[e(H_i, T_j)] \leq 1 \tag{1.2}$$

where m is the number of tokens in the hypothesis; $e(H_i, T_j)$ is the magnitude of the score of a metric comparing hypothesis token i and text token j; and α is a parameter weighting the distance penalty. $\Delta(e(H_i, T_j), e(H_{i+1}, T_k))$ measures the distance between the text constituent aligned to hypothesis token i and the text constituent aligned to hypothesis token $i + 1$. For constituents covering multiple tokens, this value is the *minimum* distance between any token covered by the constituent covering T_j and any token covered by T_k. This distance function could be measured in a variety of ways: for example, in tokens, or by edges in a path through a dependency parse tree. $I[e(H_i, T_j)]$ is an indicator function indicating that token i in the hypothesis is mapped to token j in the text.

For alignments that combine constituents of different granularities, the formulation above uses as token-level edge-weights the magnitude of the edge score for the mapped constituents covering the pair of tokens in question. For example, an edge between two Named Entities with a score of 1.0 would count as 1.0 for each token covered by the Named Entity in the Hypothesis – a Named Entity covering two indexes would therefore generate an edge with the value 2.0. This avoids penalizing matches of constituents larger than a single token.

In our own RTE system [50], we did not have alignment training data, so we selected the alignment parameter α by hand (a positive value close to zero, sufficient to break ties), and used brute force search to find the optimal alignment. The search time has an upper limit, after which a greedy left-to-right alignment is used in place of the optimal solution. We used the number of tokens as the distance measure Δ.

The search algorithm we used is shown in figure 1.12. EdgeSetList is populated as follows: for each index in the text span of the Hypothesis, all edges from constituents *starting at* that index are collected in a set, which added to the EdgeSetList. All possible alignments are considered and scored, and the highest scoring alignment returned.

The function *getNextAlignment* is used to iterate over all possible sets of edges that respect the "one edge per Hypothesis token" constraint. To do this, it uses a CounterSet: this is an object that stores the total number of edges from constituents covering each index of the hypothesis, and an index indicating which edge in the EdgeSetList for the corresponding index was used in the previous alignment. To generate the next alignment, it increments the first EdgeSet index not already at the last edge in the set of edges for the corresponding Hypothesis index. If an individual counter is at the maximum index, it is reset to the first index, and the next counter is processed. If all counters are at the maximum index, all alignments have been considered.

To generate the alignment corresponding to the current CounterSet values, the EdgeSetList is traversed. Starting from the set of edges from constituents starting at the lowest index, the edge corresponding to the index in the corresponding counter is selected. The last index of that edge's Hypothesis constituent is found, and intermediate indexes are skipped. The next index not covered by the Hypothesis constituent is then processed, and so on until the Hypothesis indexes have been traversed.

(As written, the algorithm may generate duplicate alignments when the CounterSet is incremented, but the incremented counter is in the interval covered by a constituent corresponding to an edge selected by a counter for a lower Hypothesis index. In the interests of clarity and space, the duplicate detection has been omitted. The algorithm presented is nonetheless correct – just not as efficient as it could be.)

Running Example:

In the alignment step of our LEA system, we combine the Word and Idiom views, and align the Named Entity view separately. The rationale is that we can use the same word metric for the Idiom constituent as for the Word constituents, and we believe that the idiom replacement effectively generates a new sentence, where the replacement term competes with the original idiomatic term – it doesn't make sense to partially match the idiom.

The alignment generated by the LEA system is shown in figure 1.13. LEA's implementation of the distance function shown in equation 1.1, for simplicity, always returns 0, though it is possible to specify a penalty for distance that will tend to group edges when the Text is very long, and there are multiple matching words

```
findBestAlignment( edgeSet, hypGraph, textGraph )
  bestScore ← 0.0
  edgeSetList ← ∅
  foreach index ( getIndexes( hypGraph ) )
      currentEdgeSet ← findEdgesWithStartIndex( hypGraph, index )
      edgeSetList ← edgeSetList ∪ currentEdgeSet
  counterSet ← getCounterSet( edgeSetList )
  bestAlignment ← ∅
  do
      currentAlignment ← getNextAlignment( edgeSetList, hypGraph, textGraph, counterSet )
      score ← scoreAlignment( currentAlignment )
      if ( score > bestScore )
          bestAlignment ← currentAlignment
          bestScore ← score
  while ( currentAlignment != ∅ );
  return bestAlignment

getNextAlignment( edgeSetList, hypGraph, textGraph, edgeSetCounters )
  currentAlignment ← ∅
  if ( incrementCounters( edgeSetCounters ) )
      position ← 0
      maxPosition ← sizeOf( edgeSetCounters )
      nextUncoveredIndex ← 0
      while ( position < maxPosition )
          position ← position + 1
          if ( nextUncoveredIndex <= position )
              currentEdgeSet ← edgeSetList[ position ]
              currentPositionCounter ← edgeSetCounters[ position ]
              currentEdge ← currentEdgeSet[ currentPositionCounter ]
              currentAlignment ← currentAligment ∪ currentEdge
              hypConstituentId ← getHypConstituentId( currentEdge )
              hypConstituent ← findConstituent( hypGraph, hypConstituentId )
              lastIndex ← getLastIndex( hypConstituent )
              nextUncoveredIndex ← lastIndex + 1
  return currentAlignment

incrementCounters( edgeSetCounters, edgeSetList )
  index ← 0
  while ( index < sizeOf( edgeSetList ) )
      counter ← edgeSetCounters[ index ]
      edgeSet ← edgeSetList[ index ]
      maxCount ← sizeOf( edgeSet )
      if ( counter < maxCount )
          counter ← counter + 1
          return true
      counter ← 0
      index ← index + 1
  return false
```

Figure 1.12. Algorithm for finding the best alignment for a set of views. The function "getIndexes()" returns a sorted list of word indexes for a graph.

in the Text for certain words in the Hypothesis.

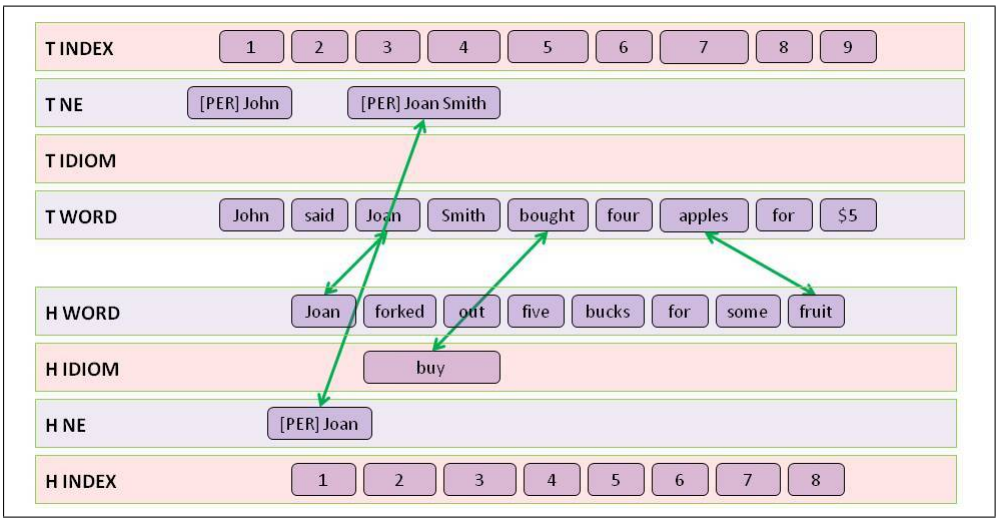

Figure 1.13. Best alignment by LEA for sample entailment pair; aligned components are connected by green arrows.

The simple LEA uses a greedy alignment approach, taking the maximum value match for each individual Hypothesis word. In the Idiom and Word view alignment, the idiom replacement counts twice, as it covers two word indexes. Function words like articles ("a", "the", etc.) and prepositions ("on", "of", etc.) generally carry much less semantic content than nouns, verbs and adjectives; LEA therefore uses a list of stopwords containing such terms and ignores their edge scores. The total alignment score for the best alignment (shown in the figure) is 0.43.

The Named Entity view is also aligned. There is only one Named Entity constituent in the hypothesis view, and it is aligned using its highest-scoring edge.

1.3.7 Inference

The Inference component of RTE systems makes the final decision about the label (and score) assigned to each entailment pair. While we present it as distinct from the Alignment step, there are approaches in which the two are closely coupled.

In some systems, inference is a very simple comparison of the alignment score to a threshold. In the two-way RTE task, if the score is higher than the threshold, the entailment pair is labeled "Entailed"; otherwise, it is labeled "Not Entailed". In the three way task, some systems perform two sequential classifications: one to distinguish between "Unknown" examples and the rest, and a second classification

step to split the rest into "Entailed" and "Contradicted" (see Wang, et al. [52]). Others apply two thresholds to the single alignment score: the second, lower threshold distinguishes between "Unknown" and "Contradicted" (see Iftene and Moruz [26]).

Other systems apply a feature extraction step after the alignment step (such as Chambers, et al. [6]). For example, these features could characterize the correspondence between dependency parse connections linking each pair of hypothesis words with the corresponding connections for the aligned text words. These features would then be used as input to a machine-learned classifier that would use them to predict the label of the entailment pair.

Some systems may alter the alignment score based on global features. Such features might be filter rules – for example, if there is a Named Entity in the hypothesis and no match is found in the text, it is very likely that the example is "Not Entailed". Other examples are negation features: usually, negations or other terms/structures affecting polarity, such as "failed to", are identified in the preprocessing or enrichment steps and encoded in the graph structure. They may then be used to affect the final decision – perhaps by switching "Entailment" to "Contradiction" if there is a negation in the Text, and none in the Hypothesis, or vice versa, when other factors indicate the Text entails the Hypothesis. In allowing metrics to return negative scores, and tracking this via edge labels, but using the magnitude of the edge score for determining alignments, such a feature is already accommodated in the proposed framework: it is possible to do the abstraction in the enrichment step, account for the enriched representation in the relevant similarity metric (by allowing it to return a negative score), then determine if negative edges are present in the final alignment.

Running Example:

The Named Entity alignment is used as a filter: if there is any Named Entity in the Hypothesis that does not match anything in the Text, LEA automatically says no. We can achieve this by thresholding the individual edge scores, and setting the predicted label to "Not Entailed" if all edges for any single Hypothesis Named Entity constituent have scores lower than the threshold. If the Hypothesis Named Entities are all matched, it consults the Word and Idiom alignment.

Since the Hypothesis contains a single Named Entity and it is aligned with a positive score to an entity in the Text, LEA does not set the label to "No Entailment", and consults the Word and Idiom alignment.

For the Word and Idiom alignment, the LEA system applies a simple threshold, as it is applied only to the two-label task. Let us assume the Word threshold is 0.67; LEA therefore predicts the label "Not Entailed" for this example based on the Word and Idiom alignment.

Note that LEA got this example wrong; to do better, it would need to be able to identify that "$5" and "five bucks" are equivalent – functionality provided by Numerical Quantity analysis and the corresponding similarity metric. Such a resource might also identify a mapping between "some fruit" and "four apples", especially if it makes use of the word similarity metric.

If there had been antonymous terms in the text and hypothesis, such as "love" and "hate", our word metric would have returned a negative score. Had there been no better (non-antonymous) match for "hate" in the text, the aligner would select the antonymous match edge because it ignores the sign of the edge value. In the inference step, the negative value would remain, and would automatically penalize the score. We could enhance the inference algorithm by changing the scoring function to use rules (such as, "if two aligned verbs from the text and hypothesis are antonymous, predict 'Contradicted"') or by making the alignment score aggregation multiplicative (a single negative edge will result in a negative overall score). Such heuristics are sometimes effective, but generally introduce new sources of error; nevertheless, such effects are taken into account and used by successful RTE systems to improve performance.

1.3.8 Training

In most successful systems, the alignment and/or inference components must be tuned to the entailment corpus using the development data set. In systems using machine learning components, this process is called *training*: the machine learning algorithm processes the entailment examples in the development corpus, computes relevant statistics, and generates a model of the problem based on characteristics of the inputs it receives, usually expressed as *features* – expressions or functions that take a specific part of the input and compute a value for each example.

In non-machine-learning-based components, there may be a process of tuning similarity functions using the development corpus, possibly by trial and error, or by brute force search over a parameter space.

We discuss the training procedures for some of the systems presented in the Case Studies (section 1.4).

Running Example:

For the LEA system, we need to compute the threshold used by the inference step to determine the entailment label. We do this by computing the best alignment for each example in the development corpus, sorting the examples by alignment score, and then testing each score as a possible threshold. We pick the threshold that correctly classifies the most examples.

You may have observed that in equation 1.1, we normalize the sum of the alignment edge scores by the number of tokens in the hypothesis. We do this so that in the inference step (and training), the decision is not biased by the length of the hypothesis. (Consider, for example, two different examples, one with a hypothesis of length 4, and another of length 12. If there are four similar components for each example, we intuitively desire different entailment labels, as it is more likely the first should be labeled "Entailed" than the second.)

1.4 Case Studies

In this section we present a summary of a number of state-of-the-art systems as case studies. For each case, we define the key characteristics of the approach, the preprocessing modules used, and the method used to predict the entailment decision (where relevant). A number of open-source resources are used by multiple systems; rather than give multiple, repeated citations for each such resource, we simply name them here, and collect all this information at the end of the chapter (see section 1.6). Our goal here is to describe interesting research in RTE, and to relate the different approaches to our framework. For specific details of implementation, we refer readers to the original publications.

Note that where possible, we have included here systems that were evaluated on the RTE 5 dataset. However, some interesting systems were only evaluated on earlier RTE data sets, so their accuracy results are not directly comparable.

1.4.1 Extracting Discourse Commitments

Hickl et al. [24] propose a framework for recognizing textual entailment based on extraction of implicit beliefs or discourse commitments. The assumption is that the text consists of many simpler constructs that are true even if the particular text-hypothesis pair does not entail. Fig. 1.14 shows a sample entailment pair with all discourse commitments; the block diagram of the system is shown in Fig. 1.15.

The preprocessing step includes syntactic parsing and semantic-dependency parsing, named entity recognition, coreference resolution, and numeric quantity recognition. The outputs of these systems are unified in a single graph representation.

In the enrichment step, the Text and Hypothesis sentences are decomposed into sets of simpler sentences that are themselves true, irrespective of the truth value of the pair. A relation extractor is used to recognize known relations, such as owner-of, location-near, employee-of, etc.; and supplemental expressions, such as parenthesis, as-clauses, and appositives.

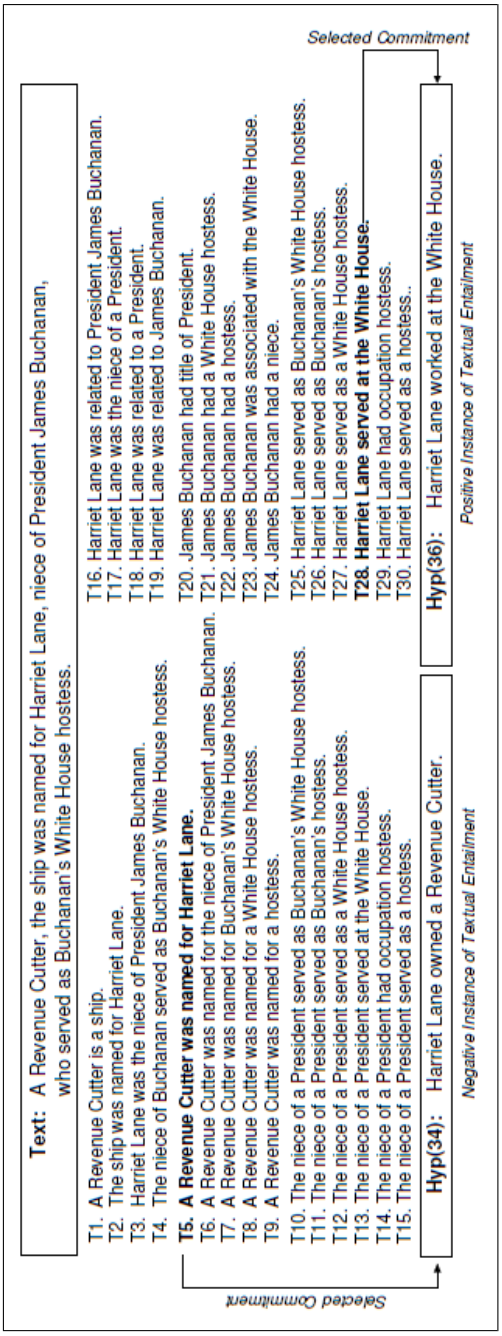

Figure 1.14. Example of discourse commitments from text. [24]

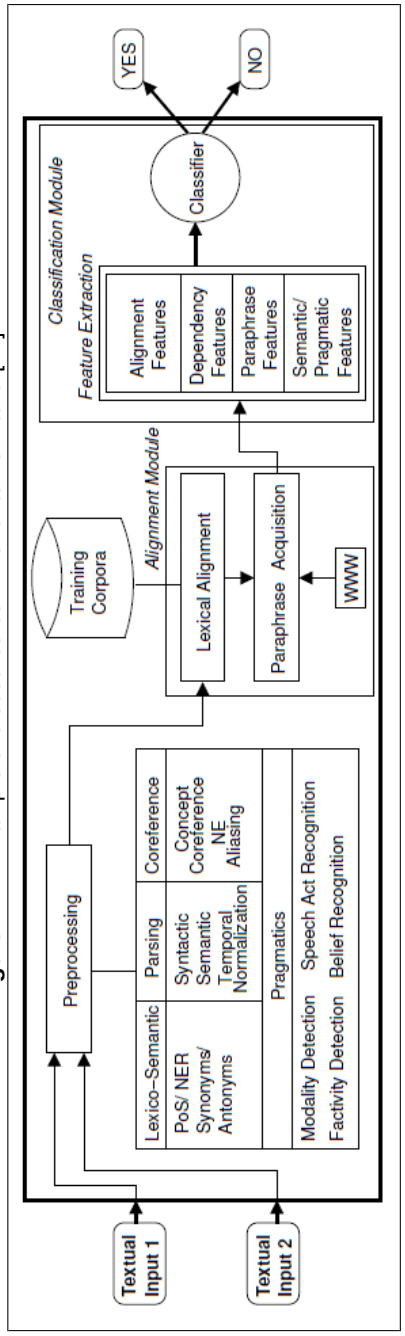

Figure 1.15. Textual entailment framework, as described in [21]

In the alignment step, a token-based aligner is applied that uses multiple similarity metrics such as WordNet-based word similarity, Levenstein string-edit distance, and named entity similarity (equality) metrics. These metrics are used to align words from hypothesis commitments to text commitments.

In the inference step, the system extracts features based on entity and argument matches, and a decision-tree classifier is used to decide if a commitment pair represents a valid entailment instance. The classifier is trained in the standard way, using features extracted for each example in the development corpus.

The system achieves an accuracy of 80.4% on the RTE 3 Test set, and a modified version of the system scored 74.6% on the RTE 4 data set (see Hickl [23]). While the performance of this system is very strong, it depends on a large corpus of proprietary supplemental training data, and the preprocessing tools it uses are also mostly proprietary. However, the underlying concept is similar to numerous other approaches, breaking the surface text down into simpler units and matching these, rather than the original words and sentences.

1.4.2 Edit Distance Based RTE

To the best of our knowledge, Tree Edit Distance (typically based on dependency parse structure) was first used for textual inference by Punyakanok et al. [46] to select answers in the task of Question Answering. Several teams later applied Tree Edit Distance to the task of Recognizing Texutal Entailment (for example, Kouylekov and Magnini [27] in RTE 1).

Mehdad et al. [36] proposes an open-source framework for textual entailment called the Edit Distance Textual Entailment Suite (EDITS) [37], which provides a basic, customizable framework for systematic development and evaluation of edit distance-based approaches to RTE. The framework allows the computation of edit distance to transform the text into the hypothesis using edit operations at the string-, token-, and tree-level. In addition, it allows specification of entailment and contradiction rules that associates a score with the transformation rule of an element from the text to an element from the hypothesis.

The EDITS framework also defines a common text-annotation format to represent the input text-hypothesis pair and the entailment and contradiction rules. The training data is used to learn a distance model. The EDITS workflow is shown in Fig. 1.16.

In the system submitted to TAC RTE 5, the preprocessing step used dependency parsing, part-of-speech tagging, lemmatization, and morphological analysis.

The graph generation and alignment steps are integrated. The lowest cost edit distance is determined using a set of operations (insertion, deletion, and substitution), each of which has an associated cost. These costs are learned using an op-

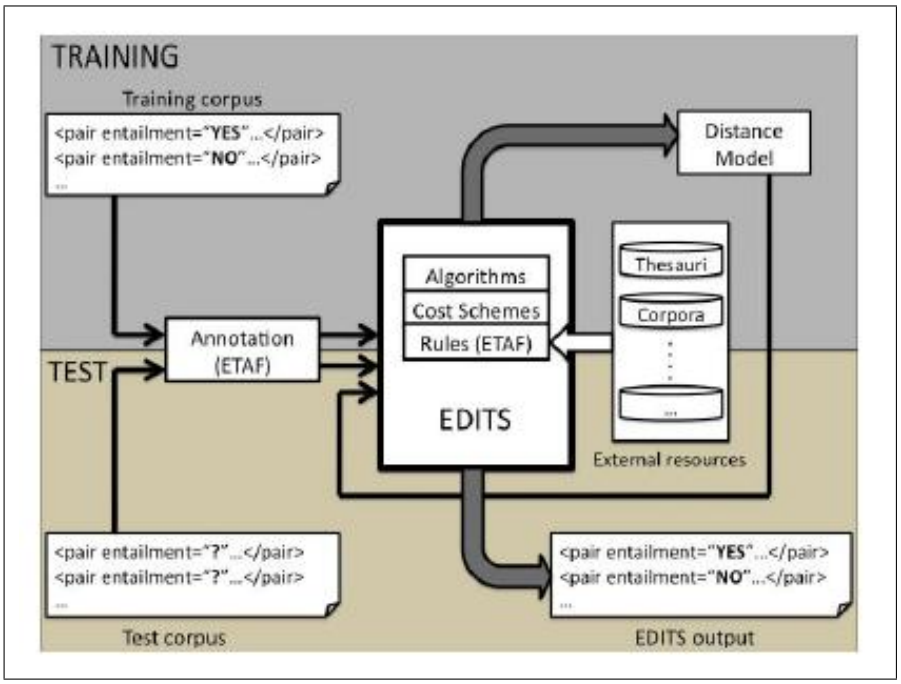

Figure 1.16. EDITS workflow, as described in [37]

timization algorithm, together with a threshold score that maximizes performance on the development set. Word-level substitution resources were derived from VerbOcean [9], WordNet [19], and Latent Semantic Analysis of Wikipedia.

The inference step compares the computed edit distance with the learned threshold score: if the pair's edit distance is greater than the threshold, the system assigns the label "Not Entailed"; otherwise, it assigns the label "Entailed".

The EDITS-based RTE system achieved a score of 60.2% in RTE 5, but could probably be improved by investigating new substitution resources, and possibly by enriching the input structures with e.g. Named Entity information (and using a specialized similarity measure in the inference step).

1.4.3 Transformation-based Approaches

Braz et al. [**?**] describe an RTE system based around the augmentation of a graph-based representation of the entailment pair's Text and Hypothesis using hand-coded rules designed to capture alternative expressions of information at the lexical, phrasal, syntactic, and predicate-argument levels. They provide a model-theoretic justification of their approach: when a rule is applied to the entailment

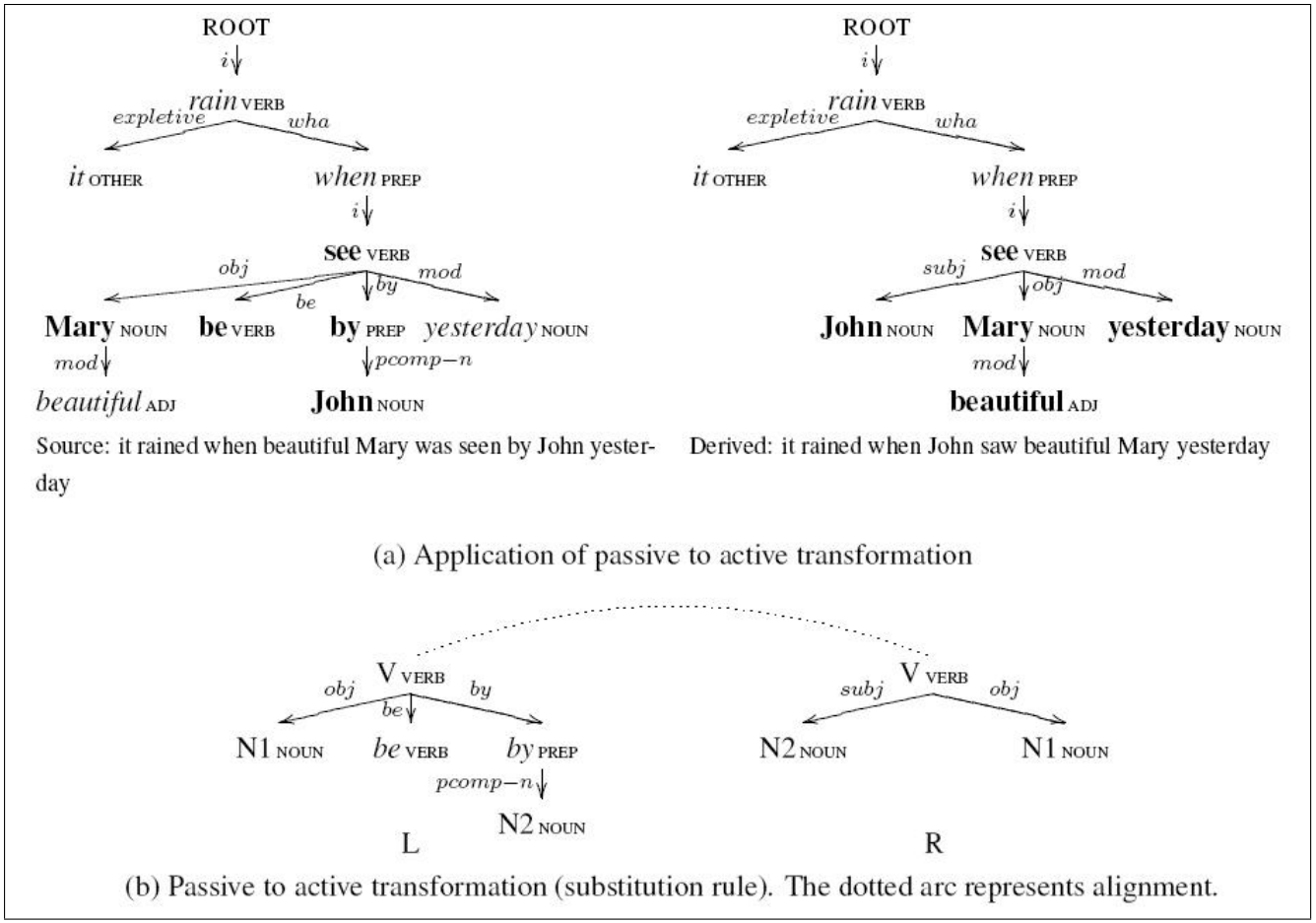

Figure 1.17. Example of an application of inference rules, as given in Bar-Haim, et al. [1]

pair Text, the augmented representation makes explicit one possible (valid) interpretation of that Text (ideally, in a way that makes the Text more closely resemble the Hypothesis, when the Text entails the Hypothesis). If any such representation of the Text subsumes the Hypothesis, the Text entails the Hypothesis.

The subsumption is formulated as an Integer Linear Programming problem, which is used to find a minimum-cost subsumption of the Hypothesis by the Text. Rules have associated costs, and these costs are further weighted depending on the level of the representation at which the rule is expressed (the intuition being that it is more important to match relations – and therefore verbs – than individual terms like determiners).

The preprocessing step of this system annotates the entailment pair with shallow parse, syntactic parse, Named Entity, and Semantic Role Labels. The enrichment step attempts to match the left hand side of each rule to the Text graph; if the rule matches, the right hand side of the rule is used to augment the Text graph. Several iterations are run, allowing limited chaining of rule applications.

There is no explicit alignment step; the inference step formulates the ILP problem and determines the minimum cost of subsumption of the Hypothesis by the Text. If the cost is too high, the entailment pair is labeled "Not Entailed"; otherwise, it is labeled "Entailed". This system was shown to outperform a smart lexical baseline on a subset of the RTE 1 development set, and achieved an accuracy of 56.1% on the RTE 1 test set (the two best systems both achieved accuracies of 58.6%.)

Bar-Haim et al. [1] describe a framework for transforming a syntactic-parse-based representation of the entailment pair Text, using rules expressed as fragments of syntactic parse trees. Hand-coded rules are used to abstract over a range of syntactic alternations. The rules pair two syntax tree fragments with placeholders representing subtrees that remain unchanged in the transformation. An example is given in Fig. 1.17.

In the enrichment step of their RTE process, the rules' heads are compared to the structure of the Text. If they match, a new syntactic parse tree is generated with the rule body; the subtrees in the original Text structure identified by the rule placeholders are copied to the corresponding positions in the new parse tree.

The inference step extracts features from the most closely matching pair of Text-Hypothesis representations (as defined by a distance metric), and these are used by a classifier to predict the entailment label. To train the classifier, the same steps are run, the features extracted, and the feature representation of each entailment pair together with the pair's label are used in the standard supervised Machine Learning paradigm. A version of this system achieved an accuracy of 60.5% on RTE 4 (Bar-Haim et al. [**?**]).

The disadvantage of these approaches is the need for many rules to capture

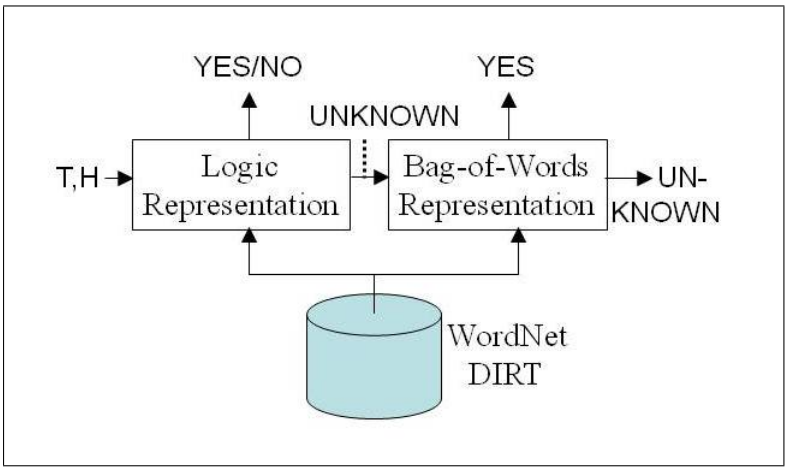

Figure 1.18. BLUE system architecture [10]

a large range of possible syntactic alternations; the high cost of producing such rules by hand makes such an effort problematic. However, the straightforward mechanism for incorporating world knowledge is appealing, as the problem of incorporating background knowledge must be overcome to make significant progress in RTE.

1.4.4 Logical Representation and Inference

The Boeing Language Understanding Engine (BLUE) system by Clark and Harrison [11] is based on a formal logical approach to RTE. It transforms the text into a logic-based representation and then tries to infer the hypothesis using a theorem-prover over this representation.

The BLUE system consists of a two-stage pipeline, as shown in Fig. 1.18. Initially, the text and hypothesis are parsed into a logical representation, using a bottom-up chart parser [22]. The logical form is a simplified tree structure with logic-type elements. It incorporates some pre-processing steps such as dependency parsing, POS tagging, and pronoun and reference resolution. Modality attributes, such as plurality, tense, negation, etc. are represented by special predicates in the logical form. This logical representation is used to infer entailment, based on subsumption and equivalence using WordNet and DIRT Inference rules. If the logical inference step fails to decide entailment or contradiction, a bag-of-words alignment model is used (in conjunction with WordNet and DIRT inference rules) as a back-off inference module.

BLUE tries to find an explanation for the entailment decision using the logical

theorem-prover to search for a chain of reasoning from text to hypothesis. It is, however, limited by errors in the knowledge sources and the pre-processing stages such as parsing and semantic analysis. Further, according to the analysis presented in Clark and Harrison [11], the presence of some implicit knowledge in text, combined with lack of knowledge to bridge the semantic gap between the text and hypothesis, presently limits the performance of the system (61.5% on RTE 5.)

One strong positive characteristic of this system is that it produces an explanation of its label, which allows the user to identify sources of errors, and to assess the reliability of the system: if the explanation is plausible for a given set of entailment examples, we may be more confident that this system will perform well on unseen examples from a similar domain.

1.4.5 Learning Alignment Independently of Entailment

De Marneffe et al. [15] investigate alignment independently of Recognizing Textual Entailment, proposing that alignment be thought of as identifying *relevant* portions of the Text, with the idea that this is simpler than determining which portions of the text entail portions of the hypothesis. They formalize alignment as an optimization problem that accounts for alignments of individual tokens in the hypothesis and of pairs of hypothesis tokens connected by a dependency edge. They use human-annotated alignment data to train their aligner, which they evaluate in its own right. This is the basis of the alignment step in the entailment system described in MacCartney et al. [29], where it is used as a source of features for a global classifier. They present a useful formulation of alignment in terms of an objective function. One drawback of their approach is that they require annotated alignment data to train their system, which is time- and resource-intensive to produce.

MacCartney et al. [30] generalize the alignment problem to the phrase level (where *phrase* simply means *contiguous text span*), and formalize the alignment score in terms of equality, substitution, insertion, and deletion of phrases in the Text with respect to the Hypothesis. They train this model using lexical alignment labelings generated by Brockett [4]. While they report an improvement over two lexical-level alignment baselines, they did not observe significant differences in performance between the phrase-level system and a token-level alignment by the same system (i.e., where the phrase size is fixed at one token). One limitation of this approach is that it appears to disregard known constituent boundaries and does not seem to offer a clean mechanism for applying specialized similarity resources in ways other than uniformly across all contiguous text spans. Moreover, it requires labeled alignment data, of which only a limited amount is available, and that too only at the token level. However, their solutions to the problem of training an

aligner and of exploring the possible space of alignments at run-time are elegant and clearly described.

1.4.6 Leveraging Multiple Alignments for RTE

Two difficulties faced by RTE system developers who wish to use deeper NLP analytics are the integration of NLP analyses operating at different granularities (word, phrase, syntax, and predicate-argument levels), and the application of similarity metrics or other knowledge resources (such as rules) in a consistent way across these different layers of representation. In both alignment- and global similarity-based approaches to RTE, problems arise when trying to incorporate multiple knowledge resources, as resources developed for different tasks may have incompatible outputs even when they all return real-valued scores. For example, a Named Entity metric may return a score of 0.6 that indicates relatively low similarity, while a WordNet-based metric may return the same value to indicate relatively high similarity: their scores are not *compatible*, because the same returned score does not have an equivalent meaning.

Sammons et al. [50] attempt to address both these problems, describing a multi-view approach in which different sources of NLP analysis are represented in separate views of the data, though comparable levels of representation may be combined in the same view. Specialized knowledge resources are encoded as metrics operating on these individual views. Their system uses multiple alignments of the Text and Hypothesis in each entailment pair, separating views with incompatible metrics into separate alignments.

Features are defined over individual alignments and also between alignments, based on the observation that (for example) if lexical-level alignments or Semantic Role-based predicate-argument structure alignments indicate entailment, but alignments using Numerical Quantity metrics do not, this is a good indication that the Text does not entail the Hypothesis. These features are used to train a classifier.

The multi-view, multi-alignment model allows a modular approach to integrating new NLP analytics and knowledge resources, and the machine-learning based inference component allows the system to determine the reliability of cues from different sources of analysis. The system performs competitively with other alignment-based systems, scoring 66.6% on the RTE 5 two-way task.

1.4.7 Natural Logic

MacCartney and Manning [31] propose a framework based on a natural logic-based representation and inference process to address the textual entailment challenge. In this approach, valid inference patterns are characterized in terms of syntactic

forms that are close to the original surface form, without involving the full semantic interpretation.

The underlying idea is to break down the entailment process into a sequence of smaller entailment decisions, whereby portions of the Text are compared to portions of the Hypothesis and related by one of a closed set of operations that indicate the semantic relationship between the two. For example, semantic containment identifies when one concept generalizes another, while semantic exclusion indicates when one concept, if true, precludes the other being true.

They also classify context structures that affect the validity of a given relationship in admitting entailment of the Hypothesis by the Text. This is expressed in terms of polarity and monotonicity. Polarity must be compatible to permit entailment, and accounts for negation and modal modifiers of predicates expressed in the entailment pair. Monotonicity specifies whether a Text concept must be more general or more specific than its counterpart in the Hypothesis, and typically arises in specific types of construction such as universally quantified statements.

To determine entailment, the text is first represented as basic semantic relations (premises), and then a sequence of edit operations is applied to transform the premise to the hypothesis. For each edit operation, a lexical entailment relation is predicted using a statistical classifier, and these relations are propagated upward through a syntax tree, according to semantic properties of intermediate nodes. The final step composes the resulting entailment relations across the edit sequence.

This approach works well on simple sentences, such as those in the FraCaS corpus (Cooper et al., [12]), but it becomes much harder to reliably extract the basic premises from the Texts in entailment pairs, as world knowledge is often required to infer relations more closely reflecting the structure of those in the Hypothesis. To apply the Natural Logic inference to the RTE task, Padó et al. [42] combine the alignment system described above with a simple NatLog edit distance via a straightforward linear function, and achieve a score of 62.7% on RTE 4.

1.4.8 Syntactic Tree Kernels

The SemKer system, proposed by Zanzotto, et al. [34], uses syntactic tree kernels to define similarity between pairs of text trees and pairs of hypothesis trees drawn from each pair of entailment examples, and extends the model with a similarity measure based on Wikipedia. The system uses a dependency tree based representation, with abstraction of nodes via lexical/semantic match. SemKer computes the similarity between terms using the Syntactic Semantic Tree Kernel (SSTK) [3], which encodes lexical similarity in the fragment (subtree) matching.

The system has a preliminary lexical alignment stage, which establishes potential subtree matching locations, called “anchors”. These focus the application

of the subtree matching component, which determines the final alignment between text and hypothesis for each entailment pair.

To train the inference model, these anchors are then abstracted into generic placeholders, and a second tree-kernel-based similarity function is applied to compare patterns of alignments between entailment pairs. The goal is to learn more general structural correspondences that apply over multiple entailment pairs. A Support Vector Model is trained using this inter-pair distance metric and the entailment example labels; this model is applied at the inference step of their RTE system.

The system performed well on RTE 5, with 66.2% accuracy on the two way labeling task (one of the top-5 scores). To capture the large variety of syntactic variations permitted in natural language text, and thereby improve its performance and capacity to generalize, this approach appears to need a lot more training data. It would be interesting to see how it performed if it were trained using the proprietary corpus described by Hickl [23].

1.4.9 Global Similarity using Limited Dependency Context

Iftene and Moruz [26] developed the system that performed best in both the two- and three-way entailment task in RTE 5. The structure of their system, like that of many other successful systems, closely matches the one we have described in section 1.3.

In the preprocessing step, the text of the entailment pair is first normalized to expand contractions (e.g., "is not" instead of "isn't") and replace some punctuation characters. This improves the performance of the off-the-shelf packages they use. The induced representation of the entailment pair is based on a dependency parse tree, enriched with Named Entity information. The preprocessing step also applies some custom resources that annotate specific relations (such as "work-for"), numerical quantities, and languages.

The alignment step comprises local and global scoring functions. First, each hypothesis constituent is mapped to the best candidate text constituent. This process includes the application of rules derived from WordNet, Wikipedia, VerbOcean, and other custom resources to identify possible mappings between dissimilar text/hypothesis term pairs; these mappings have associated scores. These local fitness scores also account for the parents of the nodes being compared, and the types of dependency edges connecting them.

These local alignment scores are then integrated, and some adjustments are made based on global characteristics of the alignment, such as whether the Named Entities in the Hypothesis are matched by Entities in the Text, and whether an aligned predicate is negated in one of the Text and Hypothesis but not the other.

The inference step applies two thresholds to the resulting score: a higher threshold that distinguishes between "Entailed" and "Not Entailed", and a lower threshold that distinguishes between "Unknown" and "Contradicted". These thresholds are tuned to maximize performance on the three-way task for the Development set; the two-way labeling score is derived directly from the three-way labeling by combining the "Unknown" and "Contradicted" labels to generate "Not Entailed" labels.

This system achieved accuracies of 68.5% on the RTE 5 three-way task and 73.5% on the RTE 5 two-way task.

1.4.10 Latent Alignment Inference for RTE

Chang, et al. [8] develop a joint learning approach that learns to make entailment decisions along with learning an intermediate representation that aligns Texts and Hypotheses. No supervision is assumed at the intermediate alignment level. They propose a general learning framework for RTE and other problems that require learning over intermediate representations.

The framework uses the declarative Integer Linear Programming (ILP) inference formulation (see Chang et al. [7]), where the intermediate representation can be easily defined in terms of binary variables and knowledge can be injected as constraints in the model. The model assumes that all positive examples have *at least one* good intermediate representation (alignment), while negative examples have *no* good intermediate representation. During training, if the model generates a "good" (valid) alignment – in the sense that the resulting entailment decision based on the features activated by this alignment is correct – the learning stage uses this as a positive example for the entailment classifier and also to provide feedback to the alignment model.

The text and hypothesis are represented as graphs, where the words and phrases are nodes and dependency relations between words form the edges. In addition, directed edges link verbs to the head words of their semantic-role-labeled arguments. The mappings between the nodes and edges in the text graph and the hypothesis graph define the alignment. These alignment variables are constrained using relations between word mappings and edge mappings: for instance, an edge mapping is active only if the corresponding word mappings are active.

One key aspect of this approach is that the alignment step is not specified as a separate, stand-alone task; rather, a space of alignment structures is defined, and the gold standard training labels of the target application are used together with an optmization approach to determine the *optimal intermediate representation* for the *target* task, i.e. the representation that maximizes performance on the target task. This obviates the need for expensive annotation efforts on the intermediate

structure.

Chang, et al. [8] apply their framework to transliteration discovery, paraphrase identification, and recognizing textual entailment. For the RTE task, the preprocessing step uses Named Entity, dependency parse, Semantic Role Labeling, and Coreference analysis and collapses them into a single, canonical graph structure. The graph generation step uses similarity metrics for words and Named Entities (see Do et al. [18]) but also computes alignment edges between edges in the Text and the Hypothesis, where the edges' sources and sinks are also aligned.

The alignment and inference step are integrated, with the optimal alignment and optimal entailment decisions based on the feature weights learned in the training process. Chang et al.'s system achieved an accuracy of 66.8% in the 2-way task for the RTE 5 corpus.

1.5 Taking RTE Further

The results in figure 1.2 show that there is still a long way to go before RTE can be considered a solved problem.

From the various examples given throughout this chapter, it should be evident that reliably recognizing textual entailment requires many smaller entailment phenomena to be handled – such as identifying when two strings refer to the same underlying entity, or applying background knowledge to infer something not explicitly stated in the text. In this section, we present some particularly important capabilities that are not yet (sufficiently) developed, to provide a possible focus for ongoing research.

1.5.1 Improve Analytics

All successful RTE approaches depend on input from other NLP tools. The more complex the annotation, the poorer the performance of the corresponding tool tends to be. Improving the performance of resources such as Named Entity Recognizers and Syntactic Parsers will tend to improve the performance of RTE components that depend on them. This is particularly true of RTE systems like that of Bar-Haim, et al. [1] that enrich the input using rules based on parse structure.

One functionality commonly identified as crucial to textual inference is Co-reference Resolution. While co-reference systems achieve reasonable performance on purpose-built corpora, they (like other NLP applications) tend to perform significantly less well on raw text from other domains. This is partly due to over-fitting to the evaluation domain, and partly due to assumptions made in the evaluations themselves. In particular, systems perform badly linking co-reference of phrasal

(i.e., non-pronominal) mentions to the correct entity.

1.5.2 Invent/Tackle New Problems

There are many linguistic phenomena that seem relevant to RTE but have no existing NLP resource; they may not even be widely recognized as necessary tasks by the NLP community. Such problems may lack relevant corpora, even if they are recognized as potentially useful.

One example that seems particularly relevant is trace recovery: identifying places in sentences where the writer implicitly refers to something, relying on the reader to fill in a gap based on the context: for example, in the sentence "John sold apples, Jane oranges.", a human reader infers that the "sold" relation holds between "Jane" and "oranges". Attempts at training syntactic parsers to recover traces, such as Dienes and Dubey [17], have had only limited success, in part because syntactic parsers are not error free, and in part because the original annotation (see Marcus, et al. [32]) is not consistent.

A related problem is Zero Anaphora resolution. For example, in the sentence "One rainy day is bad enough, but three in a row are intolerable", a human reader recognizes the "three" as referring to "three rainy days". There are publications addressing this problem, but so far, no application has been made available that has been widely used by the community.

NLP tools typically tag only explicit content, and significant additional processing is required to solve the problems described above. If these problems were solved – for example, by identifying the places where content is missing, or better yet by adding the missing content – some NLP analytics would generate more useful output, from the perspective of RTE and also of other NLP tasks.

Another topic deserving long-term attention is discourse structure. The harder RTE examples require synthesis of information spread across multiple sentences; in the search task piloted in RTE5, Mirkin, et al. [39] observed that in some news articles, information from the headline is needed throughout the article to fully understand sentences. Certain relations between events, such as causality and timing, may be expressed by structures that are not restricted to single sentences – the boundary of many NLP tools. These very long-distance dependencies are signified by discourse structure, which is very much an open topic in NLP research. With the publication of the Penn Discourse Treebank [45], there is a resource suited to developing analysis of some classes of long-range dependencies.

Text: Local health department officials were quoted as saying that the bridge over the Santa Barbara river, in southern Peru's Ayacucho province, "broke in two" as students and teachers from four rural schools were crossing it while going home. . . Local police said the 120-meter bridge, made of wooden boards and slats held together by steel cables, collapsed because too many people were on it.

Hyp: The Peruvian bridge in Ayachuco province broke because of the weight on it.

Figure 1.19. RTE5 example (development set, id 34, text truncated) requiring understanding of causal relations.

1.5.3 Develop Knowledge Resources

There are many recognized entailment phenomena that are not strongly represented in the RTE corpora, but which are clearly needed for systems to achieve Natural Language Understanding. In particular, there are certain types of reasoning that human readers do without conscious effort, but which are extremely challenging for an automated system. Some examples are causal and spatial reasoning (examples from the RTE5 corpus are presented in figures 1.19 and 1.20.

Causal reasoning relates to world knowledge a human can bring to bear that expresses domain-specific cause-and-effect relationships: for example, that bombs can explode, and that explosions can cause injury and/or death to people.

In the entailment pair in figure 1.19, a reader must infer that because people exert force (weight) on structures they stand on, and that too much weight on a bridge implies too many people; it is therefore valid to conclude that the cause of the bridge collapse can be expressed in the Hypothesis as a result of "too much weight" instead of "too many people".

In the entailment pair in figure 1.20, the text states that political leaders in Baghdad and Washington are concerned about bombings, and then gives details of three bombings. The reader must infer that "south of Baghdad" implies "in the Baghdad area", and that Abu Gharib, if only by virtue of being located in Iraq (which itself might be known via background geographical knowledge), can also be considered to be "in the Baghdad area", at least in the given context.

Other types of reasoning are less generic, but seem well represented in NLP tasks; for example, recognizing various kinship relations in order to identify strong connections in entailment pairs like that in figure 1.21. Here, the challenge is to specify the necessary knowledge in a consistent, sufficiently unambiguous way, that is also accessible to RTE systems. The CYC database [33] is a vast repos-

Text: Three major bombings in less than a week will be causing some anxiety among political leaders in Baghdad and Washington. Last Thursday 10 people were killed by a car bomb at a crowded cattle market in Babel province, south of Baghdad. On Sunday more than 30 died when a suicide bomber riding a motorbike blew himself up at a police academy in the capital. Tuesday's bombing in Abu Ghraib also killed and wounded a large number of people - including journalists and local officials.

Hyp: Some journalists and local officials were killed in one of the three bombings in the Baghdad area.

Figure 1.20. RTE5 example (development set, id 224, text truncated) requiring understanding of spatial relations.

Text: British newsreader Natasha Kaplinsky gave birth to a baby boy earlier this morning at around 08:30 BST. She had been on maternity leave since August 21. Kaplinsky had only been working with Five News just over a month when she announced she was pregnant. Her husband of three years, investment banker Justin Bower announced "We're absolutely thrilled."

Hyp: Natasha Kaplinsky and Justin Bower got married three years ago.

Figure 1.21. RTE5 example (development set, id 224, text truncated) requiring understanding of kinship relations.

itory of knowledge, painstakingly encoded in a consistent logical form; but it is not widely used precisely because its representation constrains its use. Lin and Pantel's DIRT rules [28], however, are widely considered to be in a usable form (dependency tree paths with slots for entities), but to be too noisy to be of practical use (see Clark and Harrison [11], and the ablation study in Bentivogli, et al. [2] for some examples). The kinds of "facts" identified by OpenIE approaches like TextRunner [53] are also noisy, and have yet to be proven useful in RTE.

Noise-free sets of rules for common domains, in an appropriate representation, would be a valuable asset; Szpektor, et al. [51] propose a promising representation.

1.5.4 Better RTE Evaluation

The current evaluation of RTE focuses mainly on absolute performance, reporting the accuracy of a given system in predicting one of two labels ("Entailed" and "Not

Entailed") for the two-way task, or one of three labels ("Entailed", "Contradicted", and "Unknown") for the three-way task. The problem for RTE researchers is that from the human reasoning perspective, predicting this label requires many other entailment decisions to be made, and that the single final label does not tell us anything about the way the system handles those smaller decisions. In the example in figure 1.20, a human reader must reason that there are three bombing events reported in the Text, that the phrase "including journalists and local officials" represents entities specified by "a large number of people", and that the three separate locations mentioned in the Text are all in the Baghdad area. Without knowing what the system actually did to handle each of these problems, we cannot reliably predict how the approach used by the system will handle new entailment problems requiring similar kinds of inference: a system might incorrectly predict the entailment label, but might be reliably resolving inferences requiring spatial reasoning, for example. If reliable solutions for entailment sub-problems are developed, it is in the interest of the RTE community to recognize this and reuse the solution, to avoid duplication of effort and focus attention on other needed capabilities.

There are two obvious solutions to this problem: require systems to generate explanations of their answers, and/or annotate RTE examples with more information than the present binary or ternary label.

At least one RTE system (Clark and Harrison [11]) already generates explanations that are useful in identifying flaws in its knowledge resources, though it is strongly dependent on its formal logical inference process, which is brittle in the face of noisy inputs. But even with this, the steps in the explanation are not always clear, and it is not self-evident that the kinds of steps made by a human reasoner can all be accommodated in a transparent way in this formalism.

A standard format for explanation – and a corresponding annotation of entailment examples – would be a step forward to making it possible for RTE system builders to work on explanation generation in a systematic, coordinated way, rather than each following an independent direction.

A second option is to annotate RTE examples more fully, but without committing to a particular representation for explanations. As a partial measure, an annotation standard for determining and recording the entailment phenomena that are required to predict the entailment label for an entailment pair would allow at least an approximate understanding of which capabilities a given RTE system has, by checking the correlations between correctly labeled examples and the active entailment phenomena. In addition, such labeling would allow researchers to quickly extract entailment corpora with specific characteristics, allowing evaluation of phenomena-specific resources in the context of RTE performance. These questions are raised, and an annotation standard proposed, in Sammons, et al. [49].

1.6 Useful Resources

This section gives some information about resources used by some of the RTE systems evaluated in the RTE challenges.

1.6.1 Publications

Many RTE researchers participate in the NIST TAC RTE challenge, which publishes data sets and descriptions of participating RTE systems at its web site [12]. You can find pointers to additional research publications on RTE at the ACL RTE portal [13]. Other publications relating to RTE appear in conferences such as ACL, EMNLP, COLING, and AAAI; ACL and EMNLP papers are available online via the ACL anthology [14].

1.6.2 Knowledge Resources

The ACL RTE portal named above also has pointers to some useful knowledge resources [15], such as collections of rules, some of which are mentioned in the case studies in section 1.4. The ACL RTE portal also has several complete RTE systems available for download.

1.6.3 Natural Language Processing Packages

Some popular NLP frameworks are LingPipe [16], UIMA [17], NLTK [18] and GATE [19], though there are other publicly available frameworks. Some of these frameworks also offer NLP modules for Named Entity Recognition, Co-reference, segmentation, etc. We have also found Thrift [20] and XML RPC libraries (such as that of Apache [21]) to be useful resources for distributing NLP tools across multiple computers.

A number of research groups make NLP annotation tools available. Stanford [22] offers a POS tagger, syntactic parser, and named entity recognizer, together with

[12] http://www.nist.gov/tac/
[13] http://www.aclweb.org/aclwiki/index.php?title=Textual_Entailment
[14] http://aclweb.org/anthology-new/
[15] http://www.aclweb.org/aclwiki/index.php?title=RTE_Knowledge_Resources
[16] http://alias-i.com/lingpipe/
[17] http://incubator.apache.org/uima/
[18] http://www.nltk.org/
[19] http://gate.ac.uk/
[20] http://incubator.apache.org/thrift/
[21] http://ws.apache.org/xmlrpc/
[22] http://nlp.stanford.edu

some resources to simplify some NLP programming tasks. The Cognitive Computation group [23] offers a large suite of NLP tools, including the state-of-the-art (as of Spring 2010) Illinois- Named Entity Tagger, Co-reference resolver, Part-of-Speech tagger, Chunker (shallow parser), and Semantic Role Labeler. They have also released their Named Entity and Lexical similarity metrics (Illinois-NESim and Illinois-WNSim). They also offer Learning-Based Java (LBJ), an extension to the Java programming language, which simplifies development and deployment of machine learning techniques as integral parts of java applications, and includes some useful NLP tools such as a sentence-level and word-level segmenter. Many researchers use syntactic parsers by Michael Collins [24], Dan Bikel [25], and Eugene Charniak [26].

There are many more implementations of these and other NLP tools, and even more publications describing unpublished applications. Those listed above are a popular subset that should help you to get started.

1.7 Summary

The task of Recognizing Textual Entailment provides a general, representation-agnostic framework for semantic inference in text processing, allowing researchers to entertain a wide range of approaches to solve the problem. The approach of the Natural Language Processing community to other textual inference problems like Named Entity Recognition and Resolution has been to tackle "component" inference tasks that can be thought of as part of some unspecified, comprehensive inference process. A popular approach to RTE is to think of Recognizing Textual Entailment as a framework that integrates (subsets of) these components in a way that fills in the gaps of this overarching process; it is in this spirit that we have proposed the RTE framework described in this chapter.

We have sought to address several distinct requirements that are in tension with each other:

- The ability to incorporate an arbitrary selection of existing NLP resources, which may not be consistent in granularity (word vs. phrase vs. predicate-argument-structure), formalism, or availability across languages.
- The flexibility to accommodate developer constraints such as engineering effort and run-time complexity.

[23] `http://L2R.cs.uiuc.edu/cogcomp`
[24] `http://people.csail.mit.edu/mcollins/code.html`
[25] `http://www.cis.upenn.edu/ dbikel/software.html`
[26] `ftp://ftp.cs.brown.edu/pub/nlparser/`

- The capacity to add new NLP analytics and knowledge resources in a modular way.
- The versatility to allow developers to use a range of approaches to inference.

The concept of alignment is a natural way to think about the RTE problem as it allows the modularization of knowledge resources via a multi-view representation of enriched text in tandem with specialized, constituent-level similarity metrics. At the system level, this allows straightforward extension of the different stages to accommodate new resources.

The framework we have proposed has been designed with respect to dominant approaches to developing NLP resources in various languages, and is intended to allow development in any language for which appropriate resources are available. It also allows for a trade-off between representational expressivity and computational speed: if shallower (less structured) knowledge resources and NLP analytics are used, a simpler inference algorithm and swifter processing will result. This also accommodates users working in languages with fewer NLP resources: while sophisticated inference may be limited by availability of NLP resources, it is still possible to develop an RTE system working at a shallower level of representation.

In our survey of promising research in the field, we have illustrated different approaches to various aspects of the RTE problem including representation, application of background knowledge resources, approaches to alignment, and inference techniques. To allow readers to incorporate insights from these works into their own RTE systems, we have indicated how the execution of each approach matches the framework we have specified.

Recognizing Textual Entailment is a complex problem, and solutions require significant planning and effort. Our goal has been to provide you with the tools to quickly get started within a model that can be extended to accommodate improvements in specific sub-tasks, and a roadmap of relevant research and useful resources.

Bibliography

[1] Roy Bar-Haim, Ido Dagan, Iddo Greental, Idan Szpektor, and Moshe Friedman. Semantic inference at the lexical-syntactic level for textual entailment recognition. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 131–136, Prague, June 2007. Association for Computational Linguistics.

[2] Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernando Magnini. The fifth pascal recognizing textual entailment challenge. In *Text Analysis Conference (TAC)*, 2009.

[3] S. Bloehdorn and A. Moschitti. Combined syntactic and semantic kernels for text classification. In *ECIR*, 2007.

[4] C. Brockett. Aligning the rte 2006 corpus. Technical Report MSR-TR-2007-77, Microsoft Research, 2007.

[5] Asli Celikyilmaz, Marcus Thint, and Zhiheng Huang. A graph-based semi-supervised learning for question-answering. In *Proc. of the Annual Meeting of the ACL*, pages 719–727, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[6] Nathaniel Chambers, Daniel Cer, Trond Grenager, David Hall, Chloé Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yen, and Christopher D. Manning. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 165–170, 2007.

[7] M. Chang, L. Ratinov, and D. Roth. Constraints as prior knowledge. In *ICML Workshop on Prior Knowledge for Text and Language Processing*, pages 32–39, July 2008.

[8] Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. Discriminative learning over constrained latent representations. In *Proceedings of HLT: NAACL*, pages 429–437, 2010.

[9] Timothy Chklovski and Patrick Pantel. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, pages 33–40, 2004.

[10] Peter Clark and Phil Harrison. An Inference-Based Approach to Recognizing Entailment. In *Text Analysis Conference (TAC)*, pages 63–72, 2009.

[11] Peter Clark and Phil Harrison. An inference-based approach to recognizing entailment. In *Notebook papers and Results, Text Analysis Conference (TAC)*, pages 63–72, 2009.

[12] Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, , and Steve Pulman. Using the framework. Technical report, 1996.

[13] I. Dagan, O. Glickman, and B. Magnini. The PASCAL Recognising Textual Entailment Challenge. 3944, 2006.

[14] H.T. Dang and K. Owczarzak. Overview of the tac 2009 summarization track. In *Text Analysis Conference (TAC)*, 2009.

[15] Marie-Catherine de Marneffe, Trond Grenager, Bill MacCartney, Daniel Cer, Daniel Ramage, Chloé Kiddon, and Christopher D. Manning. Aligning semantic graphs for textual inference and machine reading. In *AAAI Spring Symposium at Stanford 2007*, 2007.

[16] Marie-Catherine de Marneffe, Anna N. Rafferty, and Christopher D. Manning. Finding contradictions in text. In *Proceedings of ACL-08: HLT*, pages 1039–1047, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[17] Péter Dienes and Amit Dubey. Antecedent recovery: Experiments with a trace tagger. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 33–40. Association for Computational Linguistics, 2003.

[18] Quang Do, Dan Roth, Mark Sammons, Yuancheng Tu, and V.G.Vinod Vydiswaran. Robust, Light-weight Approaches to compute Lexical Similarity.

Computer Science Research and Technical Reports, University of Illinois, 2010. http://L2R.cs.uiuc.edu/∼danr/Papers/DRSTV10.pdf.

[19] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[20] Sanda Harabagiu and Andrew Hickl. Methods for Using Textual Entailment in Open-Domain Question Answering. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 905–912, Sydney, Australia, July 2006. Association for Computational Linguistics.

[21] Sanda Harabagiu and Andrew Hickl. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 905–912, 2006.

[22] P. Harrison and M. Maxwell. A new implementation of gpsg. In *Proceedings of the 6th Canadian Conference on AI (CSCSI'86)*, pages 78–83, 1986.

[23] Andrew Hickl. Using discourse commitments to recognize textual entailment. In *Proceedings of the 22nd COLING Conference*, 2008.

[24] Andrew Hickl and Jeremy Bensley. A Discourse Commitment-Based Framework for Recognizing Textual Entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 171–176, 2007.

[25] Eduard Hovy. Learning by reading: An experiment in text analysis. In *Text, Speech and Dialog*, volume 4188 of *Lecture Notes in Computer Science*, pages 3–12. Springer Berlin/Heidelberg, 2006.

[26] A. Iftene and M.-A. Moruz. Uaic participation at rte5. In *Notebook papers and Results, Text Analysis Conference (TAC)*, pages 367–376, 2009.

[27] Milen Koulyekov and Bernardo Magnini. Recognizing textual entailment with tree edit distance algorithms. In *Proceedings of RTE 2005*, 2005.

[28] D. Lin and P. Pantel. DIRT: discovery of inference rules from text. In *Proc. of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2001*, pages 323–328, 2001.

[29] B. MacCartney, T. Grenager, and M. de Marneffe. Learning to recognize features of valid textual entailments. In *Proceedings of RTE-NAACL 2006*, 2006.

[30] Bill MacCartney, Michel Galley, and Christopher D. Manning. A phrase-based alignment model for natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2008)*, 2008.

[31] Bill MacCartney and Christopher D. Manning. An extended model of natural logic. In *The Eighth International Conference on Computational Semantics (IWCS-8)*, Tilburg, Netherlands, 2009.

[32] Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1994.

[33] C. Matuszek, J. Cabral, M. Witbrock, and J. DeOliveira. An introduction to the syntax and content of cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, Stanford, CA, March 2006. AAAI.

[34] Y. Mehdad, F. M. Zanzotto, and A. Moschitti. Semker: Syntactic/semantic kernels for recognizing textual entailment. In *Notebook papers and Results, Text Analysis Conference (TAC)*, pages 259–265, 2009.

[35] Yashar Mehdad and Bernardo Magnini. A word overlap baseline for the recognizing textual entailment task, 2009.

[36] Yashar Mehdad, Matteo Negri, Elena Cabrio, Milen Kouylekov, and Bernardo Magnini. Edits: An open source framework for recognizing textual entailment. In *Notebook papers and Results, Text Analysis Conference (TAC)*, pages 169–178, 2009.

[37] Yashar Mehdad, Matteo Negri, Elena Cabrio, Milen Kouylekov, and Bernardo Magnini. EDITS: An Open Source Framework for Recognizing Textual Entailment. In *Text Analysis Conference (TAC)*, pages 169–178, 2009.

[38] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312, 1990.

[39] S. Mirkin, R. Bar-Haim, E. Shnarch, A. Stern, and I. Szpektor. Addressing discourse and document structure in the rte search task. In *Text Analysis Conference (TAC)*, 2009.

[40] Shachar Mirkin, Lucia Specia, Nicola Cancedda, Ido Dagan, Marc Dymetman, and Idan Szpektor. Source-language entailment modeling for translating unknown terms. In *Proceedings of ACL/AFNLP*, pages 791–799, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[41] Homeland Security Newswire. Darpa awards BBN $30 million in machine reading project, 2009.

[42] Sebastian Padó, Marie-Catherine de Marneffe, Bill MacCartney, Anna N. Rafferty, Eric Yeh, and Christopher D. Manning. Deciding entailment and contradiction with stochastic and edit distance-based alignment. In *Text Analysis Conference (TAC)*, 2008.

[43] Sebastian Padó, Michel Galley, Dan Jurafsky, and Christopher D. Manning. Robust machine translation evaluation with entailment features. In *Proceedings of ACL/AFNLP*, pages 297–305, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[44] M. Palmer, D. Gildea, and P. Kingsbury. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106, March 2005.

[45] Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. The penn discourse treebank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*, 2008.

[46] V. Punyakanok, D. Roth, and W. Yih. Natural language inference via dependency tree mapping: An application to question answering. In submission, 2004.

[47] D. Roth and M. Sammons. A unified representation and inference paradigm for natural language processing. Technical Report UIUCDCS-R-2008-2969, UIUC Computer Science Department, Jun 2008.

[48] Dan Roth, Mark Sammons, and V.G.Vinod Vydiswaran. A Framework for Entailed Relation Recognition. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, Singapore, August 2009. Association for Computational Linguistics.

[49] Mark Sammons, V.G.Vinod Vydiswaran, and Dan Roth. "Ask not what Textual Entailment can do for you...". In *ACL*, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[50] Mark Sammons, V.G.Vinod Vydiswaran, T. Vieira, N. Johri, M.-W. Chang, D. Goldwasser, V. Srikumar, G. Kundu, Y. Tu, K. Small, J. Rule, Q. Do, and D. Roth. Relation Alignment for Textual Entailment Recognition. In *Text Analysis Conference (TAC)*, 2009.

[51] Idan Szpektor, Ido Dagan, Roy Bar-Haim, and Jacob Goldberger. Contextual preferences. In *Proceedings of ACL-08: HLT*, pages 683–691, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[52] Rui Wang, Yi Zhang, and Guenter Neumann. A joint syntactic-semantic representation for recognizing textual relatedness. In *Notebook papers and Results, Text Analysis Conference (TAC)*, pages 133–139, 2009.

[53] Alexander Yates, Michele Banko, Matthew Broadhead, Michael Cafarella, Oren Etzioni, and Stephen Soderland. TextRunner: Open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 25–26, Rochester, New York, USA, April 2007. Association for Computational Linguistics.