The Need for Accurate Alignment in Natural Language System Evaluation

Andrew Kehler*
UC San Diego

John Bear†
SRI International

Douglas Appelt‡
SRI International

As evaluations of computational linguistics technology progress toward higher-level interpretation tasks, the problem of determining alignments between system responses and answer key entries may become less straightforward. We present an extensive analysis of the alignment procedure used in the MUC-6 evaluation of information extraction technology, which reveals effects that interfere with the stated goals of the evaluation. These effects are shown to be pervasive enough that they have the potential to adversely impact the technology development process. These results argue strongly for the use of accurate alignment criteria in natural language evaluations, and for maintaining the independence of alignment criteria and mechanisms used to calculate scores.

1. Introduction

It would be hard to overestimate the influence of evaluation on current natural language processing (NLP) research and development. In contrast to the primarily qualitative methodologies that characterized research in the 1980's, purely quantitative evaluation methods now pervade all aspects of the research and development process in many areas of NLP. These roles are well summarized by Chinchor and Dungca (1995), who refer specifically to the methods used for the U.S.-government-sponsored Message Understanding Conference (MUC) evaluations of information extraction (IE) technology:

> The resulting scores [assigned by the MUC evaluation process] are used for decision-making over the entire evaluation cycle, including refinement of the task definition based on interannotator comparisons, technology development using training data, validating answer keys, and benchmarking both system and human capabilities on the test data. (p. 33)

This passage highlights three major roles an evaluation method can serve. First, the method may be used during the process of task definition, to assess interannotator agreement on proposed task specifications and revise them accordingly, and to subsequently validate the final answer keys. Second, the method may be used to drive

* Department of Linguistics #0108, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0108. E-mail: kehler@ling.ucsd.edu
† Artificial Intelligence Center, 333 Ravenswood Avenue, Menlo Park, CA 94025. E-mail: bear@ai.sri.com.
‡ Artificial Intelligence Center, 333 Ravenswood Avenue, Menlo Park, CA 94025. E-mail: appelt@ai.sri.com.

the system development process, as system developers and machine learning algorithms rely heavily on the feedback it provides to determine whether proposed system changes should be adopted. Third, the results may be used for final cross-system comparison, on which judgments concerning the adequacy of competing technologies are based. Given their pervasiveness in the entire technology development process, the need for adequate evaluation methods is of the essence, and has thus become a prominent topic for research in itself (Sparck-Jones and Galliers 1996; CSL Special Issue 1998, inter alia).

The need to serve all these roles has necessitated the development of automated evaluation methods that are capable of being run repeatedly with little time and cost overhead. Automated methods are commonly used for application tasks, including speech recognition, information retrieval, and IE, as well as for natural language component technologies, including part-of-speech tagging, syntactic annotation, and coreference resolution. Such methods generally consist of a two-step process—first, system-generated responses are aligned with corresponding human-generated responses encoded in an answer key, and then a predetermined scoring procedure is applied to the aligned response pairs.

The second of these tasks (scoring procedure) has been the focus of most previous research on evaluation. In this paper, we focus instead on the less well studied problem of alignment. The relative inattention to problems in alignment is no doubt a result of the fact that alignment is relatively unproblematic in many natural language evaluation scenarios. In evaluations of component technologies such as part-of-speech tagging and treebank-style syntactic annotation systems, for instance, a system-generated annotation is simply scored against the human-generated annotation for the same word or sentence, regardless of whether matching it to the annotation for a different word or sentence would improve the overall score assigned. Alignment is similarly trivial in evaluations of applications such as information retrieval, since the notion of document identity is well defined. Alignment in speech recognition evaluations can be a bit more complex, but constraints inherent in the methods nonetheless prohibit clear misalignments, such that a system cannot receive credit for recognizing a word from an acoustic signal that occurred several utterances later, for instance.

As the field progresses to address higher-level interpretation tasks, however, the problem of determining alignments may become less straightforward. Such tasks may require that the output contain information that is synthesized (and perhaps even inferred) from disparate parts of the input signal, making the correspondence between information in the system output and information in the answer key more difficult to recover. A case in point is the evaluation IE technology, as most prominently carried out by the series of MUCs. For a typical MUC-style IE task, a text may contain several extractable events, and thus a method is required for aligning the (often only partially correct) event descriptions extracted by a system to the appropriate ones in the answer key. It is therefore important to investigate the issues involved in the definition of alignment criteria in such tasks, and we can use the MUC experience as a basis for such an investigation.

In this paper, we focus specifically on the criterion used for alignment in MUC-6. In light of difficulties in identifying a perfect alignment criterion for the MUC-6 task, the MUC-6 community agreed upon on a rather weak and forgiving criterion, leaving the resolution of alignment ambiguities to a mechanism that sought to maximize the score assigned. While this decision may have been thought to be relatively benign, we report on the results of an extensive, post hoc analysis of a 13 1/2-month effort focused on the MUC-6 task which reveals several unforeseen and negative consequences associated with this decision, particularly with respect to its influence on the incremental system

development process. While we do not argue that these consequences are so severe that they call the integrity of the MUC-6 evaluation into question, they are substantial enough that they demonstrate the *potential* of such an alignment strategy to have a significantly adverse impact on the goals of an evaluation. It is therefore important that these lessons be brought to bear in the design of future evaluations for IE and other high-level language processing tasks.

We begin with an overview of IE tasks, systems, and evaluation, including the alignment procedure used for MUC-6. We then provide an example from the MUC-6 development corpus that illustrates properties of the alignment process that interfere with the stated goals of the evaluation. We assess the pervasiveness of these problems based on data compiled from an extended development effort centered on the MUC-6 task, and conclude that the effect of the alignment criterion is robust enough that it could potentially undermine the technology development process. We conclude that these results argue strongly for the use of strict and accurate alignment criteria in future natural language evaluations—evaluations in which alignment problems will become exacerbated as the natural language applications addressed become more complex—and for maintaining the independence of alignment criteria and the mechanisms used to calculate scores.

2. Information Extraction, MUC, and the F-score Metric

IE systems process streams of natural language input and produce representations of the information relevant to a particular task, typically in the form of database templates. In accordance with the aforementioned array of roles served by evaluation methods, the MUCs have been very influential, being the primary driving force behind IE research in the past decade:

> The MUCs have helped to define a program of research and development.... The MUCs are notable ... in that they have substantially shaped the research program in information extraction and brought it to its current state. (Grishman and Sundheim 1995, 1–2)

There have been seven MUCs, starting with MUC-1 in 1987, and ending with MUC-7 in 1997. The metrics used—precision, recall, and F-score—are probably the most exhaustively used metrics for any natural language understanding application; precision and recall have been in use since MUC-2 in 1989 (Grishman and Sundheim 1995), and F-score since MUC-4 in 1992 (Hirschman 1998). IE evaluation has thus been extensively thought out, revised, and experimented with:

> For the natural language processing community in the United States, the pre-eminent evaluation activity has been the series of Message Understanding Conferences (MUCs).... the MUC conferences provide us with over a decade of experience in evaluating language understanding. (Hirschman 1998, 282)

The MUCs therefore provide a rich and established basis for the study of the effects of evaluation with respect to its roles noted above. As previously indicated, we will focus in this paper on MUC-6, held in 1995, and exclusively on the procedure used for aligning system-generated responses with those in an answer key.

```
⟨TEMPLATE⟩
    CONTENT    ⟨succession_event⟩

⟨SUCCESSION_EVENT⟩
    SUCCESSION_ORG    ⟨organization⟩
    POST    string
    IN_AND_OUT    ⟨in_and_out⟩
    VACANCY_REASON    { DEPART_WORKFORCE,
                        REASSIGNMENT,
                        NEW_POST_CREATED,
                        OTH_UNK }

⟨ORGANIZATION⟩
    ORG_NAME    string
    ORG_ALIAS    string
    ORG_DESCRIPTOR    string
    ORG_TYPE    { COMPANY, GOVERNMENT,
                  OTHER }
    ORG_LOCALE    string
    ORG_COUNTRY    string

⟨IN_AND_OUT⟩
    IO_PERSON    ⟨person⟩
    NEW_STATUS    { IN, IN_ACTING,
                    OUT, OUT_ACTING }
    ON_THE_JOB    { YES, NO, UNCLEAR }
    OTHER_ORG    ⟨organization⟩
    REL_OTHER_ORG    { SAME_ORG,
                       RELATED_ORG,
                       OUTSIDE_ORG }

⟨PERSON⟩
    PER_NAME    string
    PER_ALIAS    string
    PER_TITLE    string
```

Figure 1
Output template for MUC-6.

Marketing & Media: Star TV Chief Steps Down After News Corp. Takeover

In the wake of a takeover by News Corp., the chief executive officer of Star TV resigned after less than six months in that post, industry executives said.

Last week, News Corp. bought 63.6% of the satellite broadcaster, which serves the entire Asian region, for $525 million in cash and stock from Hutchison Whampoa Ltd. and the family of Li Ka-shing.

At the time of the purchase, News Corp. executives said they would like the executive, Julian Mounter, to stay. However, Star's chief balked at the prospect of not reporting directly to Rupert Murdoch, News Corp.'s chairman and chief executive officer, people close to Mr. Mounter said. Both Mr. Mounter and a Star spokesman declined to comment.

It is likely that Star's new chief executive will report to either Sam Chisholm, chief executive of News Corp.'s U.K.-based satellite broadcaster, British Sky Broadcasting, or Chase Carey, chief operating officer of News Corp.'s Fox Inc. film and television unit.

Mr. Mounter's departure is expected to be formally announced this week.

Although there are no obvious successors, it is expected that Mr. Murdoch will choose someone from either British Sky Broadcasting or Fox to run Star, said a person close to News Corp.

Figure 2
Example text from MUC-6 development set (9308040024).

2.1 Task Definition

The MUC-6 task was, roughly speaking, to identify information in business news that describes executives moving in and out of high-level positions within companies (Grishman and Sundheim 1995). The template structure that MUC-6 systems populated is shown in Figure 1. There are three types of values used to fill template slots. **String** fills, shown italicized, are simply strings taken from the text, such as *CEO* for the POST slot in a SUCCESSION_EVENT. **Set** fills are chosen from a fixed set of values, such as DEPART_WORKFORCE for the VACANCY_REASON slot in a SUCCESSION_EVENT. Finally, **Pointer** fills, shown in angle brackets, hold the identifier of another template structure, e.g., the SUCCESSION_ORG slot in a SUCCESSION_EVENT will contain a pointer to an ORGANIZATION template.

Figure 2 displays a text from the MUC-6 development corpus. When a participating system encounters this passage, it should extract the information that *Julian*

	⟨TEMPLATE-024-1⟩	⟨TEMPLATE-024-1⟩
cor	CONTENT ⟨SUCC_EVENT-024-1⟩	CONTENT ⟨SUCC_EVENT-024-1⟩
	⟨SUCC_EVENT-024-1⟩	⟨SUCC_EVENT-024-1⟩
cor	SUCCESSION_ORG ⟨ORG-024-1⟩	SUCCESSION_ORG ⟨ORG-024-1⟩
cor	POST "CEO"	POST "CEO"
cor	IN_AND_OUT ⟨IN_AND_OUT-024-1⟩	IN_AND_OUT ⟨IN_AND_OUT-024-1⟩
inc	VACANCY_REASON REASSIGNMENT	VACANCY_REASON OTH_UNK
	⟨ORG-024-1⟩	⟨ORG-024-1⟩
cor	ORG_NAME "STAR TV"	ORG_NAME "STAR TV"
mis	ORG_ALIAS "STAR"	
mis	ORG_DESCRIPTOR "THE SAT BDSTR"	
cor	ORG_TYPE COMPANY	ORG_TYPE COMPANY
spu		ORG_LOCALE WHAMPOA
spu		ORG_COUNTRY UNITED KINGDOM
	⟨IN_AND_OUT-024-1⟩	⟨IN_AND_OUT-024-1⟩
cor	IO_PERSON ⟨PERSON-024-1⟩	IO_PERSON ⟨PERSON-024-1⟩
inc	NEW_STATUS OUT	NEW_STATUS IN
inc	ON_THE_JOB UNCLEAR	ON_THE_JOB NO
	⟨PERSON-024-1⟩	⟨PERSON-024-1⟩
cor	PER_NAME "JULIAN MOUNTER"	PER_NAME "JULIAN MOUNTER"
mis	PER_ALIAS "MOUNTER"	
mis	PER_TITLE "MR."	

Figure 3
Target and hypothetical outputs for the example text.

Mounter is "out" of the position of *CEO* of company *Star TV*, along with other information associated with the event. The correct results for this passage, as encoded in a human-annotated answer key, are shown in the middle column of Figure 3.

2.2 Evaluation: Alignment

The rightmost column of Figure 3 shows hypothetical output of an IE system. The first step of the evaluation algorithm, alignment, determines which templates in the system's output correspond to which ones in the key. Generally speaking, there can be any number of templates in the key and system response for a given document; all, some, or no pairs of which may be descriptions of the same event. The alignment algorithm must thus determine the correct template pairing.

This process is not necessarily straightforward, since there may be no slot in a template that uniquely identifies the event or object which it describes. In response to this problem, the MUC community decided to adopt a relatively lax alignment criterion, leaving it to the alignment algorithm to find the alignment that optimizes the resulting score. The procedure has two major steps. First, it determines which pairs of templates are possible **candidates** for alignment; the criterion for candidacy was only that the templates share a common value for at least one slot. (Pointer fills share a common value if they point to objects that are aligned by the algorithm.) This criterion often results in alignment ambiguities—a key template will often share a common slot value with several templates in the system's output, and vice versa—and thus a method for selecting among the alternative mappings is necessary. The candidate pairs are rank ordered by a mapping score, which simply counts the number of slot values the templates have in common.[1] The scoring algorithm then considers

1 The scoring software provided for MUC-6 allows for alternative scoring configurations based on slot content, including the ability to assign different weights to slots, but this was the configuration used for development and evaluation in MUC-6.

key and response template pairs according to this order, aligning them when neither member has already been mapped to another template. Ties between pairs with the same number of common slot values are broken arbitrarily. Because this algorithm is heuristic—with many combinations of alignments never being considered—the result may not be the globally optimal alignment in terms of score assigned.

In our example, there is only one template of each type in each response, and thus there are no mapping ambiguities. The only requirement is that each pair share a common slot value, which is the case, and so the algorithm aligns each as shown in Figure 3.

2.3 Evaluation: Scoring

Once the templates are aligned, the scoring algorithm performs slot-by-slot comparisons to determine errors. The leftmost column in Figure 3 shows examples of the three types of errors that the algorithm will mark. First, while our hypothetical system recognized that the correct PERSON and ORGANIZATION are *Julian Mounter* and *Star TV*, respectively, it missed the ORG_ALIAS, ORG_DESCRIPTOR, PER_ALIAS, and PER_TITLE values that appear later in the passage, resulting in four **missing** slot fills (denoted by *mis* in the left hand column). Next, it also erroneously assigned a value to the ORG_LOCALE and ORG_COUNTRY slots in the ORGANIZATION, resulting in two **spurious** slot fills (denoted by *spu*). Finally, the system got three of the set fill slots wrong—the VACANCY_REASON slot in the SUCCESSION_EVENT, and the NEW_STATUS and ON_THE_JOB slots of the IN_AND_OUT template—resulting in three **incorrect** slot fills (denoted by *inc*). The remainder of the slot fills are **correct** (denoted by *cor*).[2] Again, pointer slots are scored as correct when they point to templates that have been aligned.

The **possible** fills are those in the key which contribute to the final score, and the **actual** fills are those in the system's response which contribute to the final score:

$$POS = COR + INC + MIS$$

$$ACT = COR + INC + SPU$$

Three metrics are computed from these results: precision, recall, and F-score. Precision is the number of correct fills divided by the total number generated by the system, and recall is the number of correct fills divided by the the total number in the key:

$$PRE = \frac{COR}{ACT}$$

$$REC = \frac{COR}{POS}$$

In the example, there are 8 correct fills, 13 generated fills, and 15 possible fills in the key, resulting in a precision of 0.615 and a recall of 0.533. Note that the four missing slot fills do not figure into the precision computation, and the two spurious fills do not figure into the recall computation. F-score is determined by a harmonic mean of precision and recall (van Rijsbergen 1979):

$$F = \frac{1}{(1 - \frac{1}{1+\beta^2})\frac{1}{REC} + \frac{1}{1+\beta^2}\frac{1}{PRE}} = \frac{(\beta^2 + 1) \times PRE \times REC}{(\beta^2 \times PRE) + REC}$$

2 There is also the possibility of getting partial credit, which we will not discuss further nor include in the equations below.

In the standard computation of F-score, β is set to one (indicating equal weight for precision and recall), resulting in:

$$F = \frac{2 \times PRE \times REC}{PRE + REC}$$

For our example, we get an F-score of 0.571.

$$\frac{2 \times 0.615 \times 0.533}{0.615 + 0.533} = 0.571$$

2.4 Focus of the Paper

Before proceeding to the analysis, we take care to note that our aim is neither to provide a thorough analysis of all problematic aspects of NLP evaluation, nor to provide a criticism of the MUCs in particular. Problematic aspects of the scoring procedures used in a variety of NLP evaluations are well attested, including the existence of side effects of scoring procedures which reward behavior that may not be perfectly consistent with the goals of the evaluations that these scoring procedures serve. For instance, word error metrics used in evaluations of speech recognition technology have been criticized for the fact that they assign the same degree of credit for the recognition of all words, a policy that rewards a focus on recognizing frequent but less important words (e.g., *um*) over more important but less frequent content words. Similarly, evaluations of syntactic annotation systems that use locally oriented crossing brackets and labeled precision/recall metrics can assign high degrees of credit to cases in which the assigned structures are fairly inaccurate when viewed more globally. Likewise, there are aspects of the scoring procedure used for MUC-6 that one could question, such as the choice of slots included in each template and their corresponding definitions, the decision to weight all slots equally without regard to perceived importance, and the choice to define the templates to have hierarchical structure and give credit for slots that merely contain pointers to lower-level templates. We believe that any mechanical evaluation is likely to have such issues, and while they are very worthy of study and debate, a more detailed discussion of them would take us too far afield from the main purpose of this paper.

The focus of this paper is purposefully more narrow, being concerned only with the effects of alignment criteria on the goals of evaluation. While it is unclear as of this writing whether there will be future MUCs, evaluation-driven efforts in IE continue to be sponsored, and future evaluations of interpretation tasks of equal or greater complexity are not only likely but crucial if the field is to progress while maintaining its current focus on quantitative evaluation. Because alignment questions will almost certainly become exacerbated as the interpretation problems addressed become more complex, and because of the aforementioned pervasiveness of evaluation in the entire technology development process, the payoff in avoiding potential pitfalls in such evaluations is high. Thus, our aim is to bring lessons learned from the MUC experience to the fore so that they can inform future evaluations that, like the MUCs, are likely to be principal driving forces for research over extended periods of time.

3. The Impact of Inaccurate Alignment

In the introduction, we described several roles that the MUC-6 evaluation has played in bringing IE technology to its current state: task definition, system development, and cross-system evaluation. We focus here on its role in the system development process, where its influence has been substantial, as it has provided system developers the ability to obtain rapid feedback with which to iteratively gauge their progress on a training

	⟨TEMPLATE-024-1⟩	⟨TEMPLATE-024-1⟩
cor	CONTENT ⟨SUCC_EVENT-024-1⟩	CONTENT ⟨SUCC_EVENT-024-1⟩
	⟨SUCC_EVENT-024-1⟩	⟨SUCC_EVENT-024-1⟩
cor	SUCCESSION_ORG ⟨ORG-024-1⟩	SUCCESSION_ORG ⟨ORG-024-1⟩
cor	POST "CEO"	POST "CEO"
cor	IN_AND_OUT ⟨IN_AND_OUT-024-1⟩	IN_AND_OUT ⟨IN_AND_OUT-024-1⟩
inc	VACANCY_REASON REASSIGNMENT	VACANCY_REASON OTH_UNK
	⟨ORG-024-1⟩	⟨ORG-024-1⟩
inc	ORG_NAME "STAR TV"	ORG_NAME "NEWS CORP."
mis	ORG_ALIAS "STAR"	
mis	ORG_DESCRIPTOR "THE SAT BDSTR"	
cor	ORG_TYPE COMPANY	ORG_TYPE COMPANY
	⟨IN_AND_OUT-024-1⟩	⟨IN_AND_OUT-024-1⟩
cor	IO_PERSON ⟨PERSON-024-1⟩	IO_PERSON ⟨PERSON-024-1⟩
inc	NEW_STATUS OUT	NEW_STATUS IN
cor	ON_THE_JOB UNCLEAR	ON_THE_JOB UNCLEAR
	⟨PERSON-024-1⟩	⟨PERSON-024-1⟩
inc	PER_NAME "JULIAN MOUNTER"	PER_NAME "RUPERT MURDOCH"
inc	PER_ALIAS "MOUNTER"	PER_ALIAS "MURDOCH"
cor	PER_TITLE "MR."	PER_TITLE "MR."

Figure 4
Target and actual outputs for the example text.

corpus. In essence, a developer can incorporate a new processing strategy or data modification, evaluate the system with the change, keep it if end-to-end performance (i.e., F-score) improves, and withdraw it if it does not. Often changes have unanticipated results, and thus a formal evaluation method is required to affirm or deny developers' (often misleading) intuitions. Likewise, the method has an analogous role for supporting systems that learn rules automatically. In a typical learning scenario, an automated procedure iteratively proposes rule modifications and adopts them only in those cases in which an objective function—that is, the evaluation method—indicates that performance has improved. Thus, it is absolutely crucial that both positive and negative system changes are reflected as such in the feedback provided by the scoring mechanism.

As we illustrate with the passage shown in Figure 2, the weak alignment criterion used in MUC-6 causes the scoring mechanism to not respect this requirement.[3] The answer key in Figure 4 is as it was in Figure 3, representing the event of Julian Mounter leaving the position of CEO at Star TV. The results of an IE system (in this case, a slightly modified version of what SRI's FASTUS [Appelt et al. 1995] extracted for this example) are shown in the righthand column. The IE system made two significant mistakes on this text: It failed to extract any information relating to the correct event, and it overgenerated an incorrect "event," specifically Rupert Murdoch's becoming CEO of News Corp.[4] Thus, the system's precision should be $\frac{0}{13} = 0$, since it generated

3 In this section we will be arguing our point primarily on the basis of a single illustrative example, where in fact the MUC development process utilized a 100-text corpus. The arguments extend to this broader setting, of course, a topic to which we return in Section 4.

4 An anonymous reviewer points out that because no slot in the template structure can be guaranteed to identify an object, intuitions alone are not enough (despite how strong they might be in a case such as this) to establish that the system output in Figure 4 actually represents a different event, rather than the event described in the output in Figure 3 corrupted by the selection of wrong names for certain slots. The FASTUS system produces byte offsets to tie information in templates to the places in the text from which they were created, and from this we can confirm that the event was indeed created solely from textual material unrelated to the event described in the key. See also footnote 7.

Table 1
Approximate distribution of values for low-entropy slots.

Template Type	Slot	Total Templates	Slot Value Distribution
SUCCESSION_EVENT	VACANCY_REASON	214	REASSIGNMENT: 101 OTH_UNK: 77 NEW_POST_CREATED: 21 DEPART_WORKFORCE: 18
IN_AND_OUT	NEW_STATUS	287	IN: 129 OUT: 148 IN_ACTING: 7 OUT_ACTING: 4
IN_AND_OUT	ON_THE_JOB	287	YES: 82 NO: 137 UNCLEAR: 96
IN_AND_OUT	REL_OTHER_ORG	287	SAME_ORG: 93 OUTSIDE_ORG: 66 RELATED_ORG: 43 ⟨none⟩: 91
ORGANIZATION	ORG_TYPE	117	COMPANY: 115 GOVERNMENT: 3 OTHER: 0
PERSON	PER_TITLE	158	"Mr.": 68 "Dr.": 3 "Ms.": 3 ⟨none⟩: 84

13 spurious slot fills for an irrelevant event, and its recall should be $\frac{0}{15} = 0$, since it generated none of the 15 slots associated with the correct event.[5]

In actuality, however, the scoring algorithm aligned these templates as shown in Figure 4, since each template pair shares at least one slot value. With this alignment, the scoring procedure identifies 8 correct slot values, 5 incorrect values, and 2 missing values, resulting in a recall of $\frac{8}{15} = 0.533$, a precision of $\frac{8}{13} = 0.615$, and an F-score of 0.571. These are the same scores received for the output in Figure 3, in which a partial but mostly accurate description of the correct event was extracted.

The fact that the wholly incorrect output in Figure 4 and the largely correct output in Figure 3 receive equally good scores demonstrates a serious flaw with the evaluation method. As shown in Table 1, the distributions of values for several slots in the MUC-6 training data have relatively low entropy, and thus alignments based on a fortuitous overlap in these values are not rare.[6] For instance, the fact that the ORG_TYPE slot in ORGANIZATION templates has the value COMPANY in 115 out of 117 instances almost ensures that any ORGANIZATION template produced by a system can get matched to an arbitrary one in the key for a given document. From this, in turn, the inaccurate alignments may then cascade: Two unrelated SUCCESSION_EVENT templates can be aligned

5 Its F-score should be undefined, since the denominator in the F-score equation will be zero. For all intents and purposes, however, the F-score can be considered to be zero, in the sense that its overall contribution to the F-score assigned to the results over a larger corpus will be zero. With this in mind, for simplicity we may speak of such cases as having an F-score of zero.

6 In some cases, the sum of the counts in the rightmost column is greater than the total template count. This is because in some cases a key entry allowed for alternative slot values; matching *any* of the alternatives was sufficient for both alignment and scoring. Likewise, instances of optional slot fills were also included.

on the basis of sharing pointers to two unrelated (but nonetheless aligned) ORGANIZATION templates. Likewise, two unrelated IN_AND_OUT templates can be aligned on the basis of sharing pointers to two unrelated PERSON templates that share the value *Mr.* for the TITLE slot. Between this prospect and the three set fills for IN_AND_OUT templates in Table 1, it is highly likely that an arbitrary IN_AND_OUT template produced by a system will overlap in at least one slot value with an arbitrary one in the key, which may in turn allow the SUCCESSION_EVENTs that point to them to be incorrectly aligned.

Although the fact that the scoring algorithm is capable of assigning undeserved credit due to overly optimistic alignments is not unknown to the MUC community (see, for instance, Aberdeen et al. [1995, 153, Table 4] for a reference to "enthusiastic scoring mapping" for the Template Element task of MUC-6), we are unaware of any previous acknowledgement of, or similar example which demonstrates, the potential severity of the problem to the extent that Figure 4 does. This notwithstanding, one might still be tempted to view this behavior as relatively benign—perhaps there is no real harm done by giving systems the benefit of the doubt, along with a little undeserved credit that goes with it. While this will perhaps result in somewhat artificially inflated scores, there may be no reason to think that it would benefit one system or approach more than another, and this concession might seem reasonable considering the fact that there is likely to be no completely foolproof way to perform alignments.

However, the potential harm that this behavior manifests in terms of the technology development process—which, to our knowledge, has never been brought to light—is that it creates a situation in which uncontroversially positive changes in system output may result in a dramatically worse score, and likewise negative changes may result in a dramatically better score. Consider a (common) development scenario in which, starting from a state in which the system produces the output in Figure 4 for the text in Figure 2, it is technically too difficult to modify the system to extract the correct (i.e., Julian Mounter) event, but in which a change can nonetheless be made to block the overgenerated (i.e., Rupert Murdoch) event. After such a modification, one would expect no change in recall, since no correct output is created or removed, and an improvement in precision, since an overgenerated event is removed.

What actually happens in this example is that recall drops from 0.533 ($\frac{8}{15}$) to zero ($\frac{0}{15}$), and precision goes from 0.615 ($\frac{8}{13}$) to undefined ($\frac{0}{0}$). To circumvent comparisons with undefined values, we can suppose that there was another, independent event extracted from a different part of the text that was aligned correctly against its corresponding event in the answer key. For simplicity, we will assume that this event receives the same score as the overgenerated event: a precision of $\frac{8}{13}$ and a recall of $\frac{8}{15}$. With the overgenerated event left in, the same scores as before are obtained: $\frac{16}{26} = 0.615$ precision and $\frac{16}{30} = 0.533$ recall, resulting in an F-score of 0.571. With the overgenerated event removed, we obtain $\frac{8}{13} = 0.615$ precision and $\frac{8}{30} = .267$ recall, resulting in an F-score of 0.372. Instead of no change in recall and an improvement in precision, the reward for eliminating the overgenerated event is the same precision, a 50% reduction in recall, and a 20-point reduction in F-score. Our clear "improvement" thus has the effect of reducing performance dramatically, implicitly instructing the developer to reintroduce the rules responsible for producing the overgenerated event.

The converse scenario yields an analogous problem. Consider a situation in which a system developer can add a rule to extract at least some of the information in the correct event—producing the output shown in Figure 5, for instance—but for whatever reason cannot make a change to block the overgenerated event. We would expect this change to result in a marked increase in both recall and precision, since unlike before, information for a relevant event is now being produced. Indeed, the

```
⟨TEMPLATE-024-1⟩
    CONTENT    ⟨SUCC_EVENT-024-1⟩

⟨SUCC_EVENT-024-1⟩
    SUCCESSION_ORG    ⟨ORG-024-1⟩
    POST    "CEO"
    IN_AND_OUT    ⟨IN_AND_OUT-024-1⟩
    VACANCY_REASON    OTH_UNK

⟨ORG-024-1⟩
    ORG_NAME    "STAR TV"
    ORG_TYPE    COMPANY

⟨IN_AND_OUT-024-1⟩
    IO_PERSON    ⟨PERSON-024-1⟩
    NEW_STATUS    IN
    ON_THE_JOB    NO

⟨PERSON-024-1⟩
    PER_NAME    "JULIAN MOUNTER"
    PER_ALIAS    "MOUNTER"
```

Figure 5
Additional output for example text.

alignment algorithm will correctly align this event with the key and leave the Rupert Murdoch event unaligned, resulting in a precision of $\frac{9}{15} = .600$, a recall of $\frac{9}{25} = .360$, and an F-score of 0.450. This is the anticipated result, and would constitute a large increase over the zero F-score that the overgenerated event should have received when standing alone. However, this is a substantial reduction from the F-score of 0.571 that the overgenerated event actually receives, a change which implicitly instructs the developer to remove the rules responsible for extracting the correct event.

In these two scenarios, positive changes to system output resulted in a dramatically reduced score. The opposite situation can also occur, in which a change that reduces the quality of the system's response nonetheless receives an increased score. One can merely reverse the scenarios. For instance, in a situation in which no output is being created for the Julian Mounter event and a developer considers adding a rule that produces the Rupert Murdoch event, the rise in F-score will indicate that this rule should be kept. Likewise, starting with the incorrect output in Figure 4 together with the correct output in Figure 5, a developer might consider removing the rule responsible for creating the correct output. This would cause the F-score to rise from 0.450 to 0.571, implicitly instructing the developer to keep it removed.

Thus, in all of these scenarios, the feedback provided by the evaluation method may steer our system developer off of the path to the optimal system state. Likewise, the same effect would occur when employing automatic learning methods that use F-score as an objective function. Starting from the state of producing the output in Figure 4, for example, suppose the learning procedure could in fact propose each change necessary to get to the desired output, that is, to (i) eliminate the rules producing the erroneous output, and (ii) add rules for producing the output shown in Figure 5. These changes would result in a precision of $\frac{9}{15} = .60$, a recall of $\frac{9}{12} = .75$, and an F-score of 0.667, which is an improvement over both the zero result that the current output should receive, and the (artificially inflated) score of 0.571 it actually does receive. However, the type of incremental search process that efficiency concerns generally necessitate—one that can only perform one of steps (i) or (ii) in a single iteration and will only adopt the proposed change if it improves on its objective function—will not find this system state, since as we have seen, either move taken first would actually reduce the F-score.

To sum, the fortuitous alignments allowed by the MUC-6 evaluation method create a situation in which both positive and negative system changes may not be reflected as such in the evaluation results. While there are other properties of the evaluation that conspire to help produce these anomalous results—including the choice to score all slot fills equally without respect to importance or entropy of their distribution of values, and to score slots which contain only pointers to other templates—these only

serve to make the effects more or less dramatic than they might otherwise be. The root cause of this behavior is the alignment process: None of the foregoing behaviors would occur if the alignment criterion was such that the templates in Figure 4 were not alignable, thus producing an F-score of zero.

4. A Case Study

The foregoing examples demonstrate the pitfalls of not employing strong alignment constraints in natural language system evaluations. In the case of IE, the constraints should come as close as possible to establishing that two template representations are meant to describe the same events or objects. Just as it would be nonsensical for evaluations of part-of-speech tagging systems to give credit for a correct tag assigned to a different word, or evaluations of syntactic annotation systems to give credit for bracketings assigned to a different sentence, IE evaluations should not give credit for a template structure that represents a different event in the text. While it may be tempting to give systems the benefit of the doubt in light of the fact that alignment in IE is inherently more difficult than in these other scenarios, we have seen that the negative consequences of such a move can subvert the goals and purposes of the evaluation, and indeed the technology development process.

Having said this, a question that naturally arises is how robust this effect actually was for the MUC-6 task. Is the example shown in Figure 4 exceptional for MUC-6, and thus useful mainly for pedagogical purposes, or is it indicative of a more pervasive problem that could impact development using a larger set of training documents? It is difficult to answer this question, of course, since one cannot replay past phases of technology development. However, we do have a case study with which to investigate this question, as we have previously performed a 13 1/2-month effort in which we focused on improving performance on the MUC-6 task. Specifically, our goal was to see how far the FASTUS paradigm could be pushed by way of making as many incremental improvements as possible. We relied heavily on the MUC-6 scoring mechanism during this process, using the feedback it provided to drive our development in the manner described in Section 3. Throughout this process, we remained ignorant of the problems that we are reporting on presently.

As a result of our effort, we have a record of the output of our system as it existed in 53 distinct states of development. We can compare the feedback provided by the scoring algorithm used during the development process (henceforth, the **standard algorithm**), with the feedback that would have been received from a more accurate and restrictive alignment criterion (henceforth, the **restrictive algorithm**), which we describe in Section 4.1. We report on two types of comparison, each at the level of individual documents (Section 4.2) and the entire 100-text development set (Section 4.3). We first report on the overall effects that the restrictive algorithm has on scores for individual system states. We then report on the extent to which the two algorithms disagreed on the **direction** of the difference in performance between a pair of system states—that is, whether the changes implemented between these states had a positive or negative effect—as this is the central factor that developers and learning algorithms use to determine whether to adopt proposed system changes. As the difference between any pair of states constitutes a set of intermediate system changes that one can evaluate, our 53 distinct states provide us with $\frac{53 \times 52}{2} = 1,378$ pairs to examine.

4.1 A Stricter Alignment Criterion

As we mentioned in footnote 1, the scoring system provided for MUC-6 allows one to customize alignment criteria based on slot content. Because no slot in a template will

consistently and uniquely identify the event or object that it describes, it is difficult or impossible to design a perfect slot-based criterion. Nonetheless, there are more restrictive parameterizations that come much closer to producing only the correct alignments. We sought out the criterion that eliminates as many incorrect alignments as possible without being so restrictive that a system would be denied partial credit for correctly extracted information.

Table 1 readily suggests a principled manner in which to restrict the mapping criterion: Avoid aligning templates solely on the basis of slots with only a small set of possible values (and in particular, those which have low entropy distributions), since shared values for these provide little evidence that the two templates represent the same entity or event. Indeed, our experience confirms that the large majority of alignment errors result from a fortuitous match on one of these slots. Each of the slots in Table 1 have at most four possible slot values, whereas the set of possible values for the remainder of the slots is essentially unbounded (except possibly for the POST slot of the SUCCESSION_EVENT template; we will return to this momentarily). Thus, while it is unlikely that templates for two unrelated companies will have the same company name, it is very likely that they both will have the value COMPANY in the ORG_TYPE slot.

We therefore modified the MUC-6 alignment criterion so that any single shared value for a slot *not in Table 1* is sufficient for alignment. (All other aspects of the alignment criterion and procedure remained unchanged.) This criterion is still generous in some cases, for instance, the system output shown in Figure 4 receives an F-score of 0.143: The system will get credit for the coincidentally identical POST slot values and for the pointers to the SUCCESSION_EVENTS that will be aligned on the basis of those values. However, a criterion that bars such alignments may also disallow certain cases in which a system should arguably deserve credit.[7] While a small amount of undeserved credit may therefore remain, this amount is dramatically reduced from that which results from the standard algorithm. All slot values were still counted for scoring purposes, as in MUC-6.

4.2 Effects of Stricter Alignment on Individual Document Results

With our more restrictive alignment algorithm in hand, we begin by looking at its effect on performance at the document level. Only 53 of the 100 MUC-6 development texts were relevant, and because the recall of an irrelevant document is undefined regardless of what the system produces, only these 53 have defined F-scores.

4.2.1 Effect on Scores. The restrictive algorithm tended to assign lower scores than the standard algorithm, as one would expect, since the best-scoring alignment found by the standard algorithm may be correctly disallowed by the restrictive algorithm. In

7 As indicated in footnote 4, it is theoretically possible that a template produced by a system and a template in the answer key originate from a description of the same entity or event, but in which the system's template is so corrupted by inaccurate or incomplete processing that the only correct slots that remain are included in Table 1. In our extensive analyses of system results, we found the number of such cases to be quite infrequent, and overwhelmed by the number of cases in which alignments based only on these slots were demonstrably incorrect. One could argue about whether any partial credit is actually deserved in the former set of cases; in any case, we believe that not assigning the small amount of credit a system would receive is a small price to pay for rectifying the much greater negative effects of maintaining an overly lax alignment strategy. Even if such partial credit is deemed deserved, however, our analyses suggest that the missed credit is more than made up for by the undeserved credit resulting from fortuitous matches on the POST slots of SUCCESSION_EVENT templates that our new criterion still allows, as described above. In fact, the overall effect of both appears to be rather negligible alone, and even more so when their opposite effects on scores are taken together.

Madison Group Says Board Has Dismissed Lucas as Its President

Madison Group Associates Inc. said its board dismissed Kenneth Lucas, president, naming Dean J. Trantalis as interim president.

The company also said two new directors – Roland Breton and Steve Gibboney – had been appointed to its board.

Mr. Lucas became chief executive of the media concern less than two months ago, when William T. Craig resigned from his job as a director and chief executive officer.

The company gave no reason for Mr. Lucas's dismissal. Neither he nor Madison executives could be reached for comment.

The management change is the latest in a series of events that have shaken the company in recent months. As previously reported, the Securities and Exchange Commission contacted several individuals about their dealings with the company. One of those individuals said the SEC had asked about how the company valued its assets.

Those assets consist largely of video libraries. According to a recent securities filing, an accountant formerly hired by Madison recommended that an independent specialist be hired to evaluate the video libraries.

Figure 6
Example text from MUC-6 development set (9403100087).

our system runs, the scores for an average of 21.2 of the 53 documents were reduced. Thus, in a typical run, a substantial percentage of the document results—40%—had benefited from incorrect alignments from the standard algorithm. The magnitude of the reduction in document scores ranged from 0.14 to 36.84 points of F-score, averaging 7.74.

Interestingly, there were also cases in which the restrictive algorithm actually assigned a *higher* score to the results for a document than the standard algorithm. One might wonder how this could happen, since the set of possible alignments allowed by the restrictive algorithm is a strict subset of those allowed by the standard algorithm. The reason lies in the fact that the alignment procedure does not perform an exhaustive search; instead, it uses the heuristic search method described in Section 2. It is therefore possible that an optimal but nonetheless correct solution exists which the standard algorithm does not find, but which the restrictive algorithm finds within the narrower search-space associated with its more restrictive criterion.

Figure 6 shows an example from the MUC-6 development corpus, and Figures 7 and 8 show a fragment of the alignments produced by the standard and restrictive algorithms, respectively. All of the remaining system output not shown received the same alignment by both algorithms. William Craig was represented by templates ⟨PERSON-087-5⟩ in the key and ⟨PERSON-087-8⟩ in the system response, and Kenneth Lucas was represented by templates ⟨PERSON-087-1⟩ in the key and ⟨PERSON-087-20⟩ in the system response; both were aligned correctly by both algorithms.

The alignment in Figure 7, along with the remainder of the output not shown, results in an F-score of 69.44 for the text. Template ⟨IN_AND_OUT-087-1⟩ is aligned with ⟨IN_AND_OUT-087-1⟩, and ⟨IN_AND_OUT-087-4⟩ is aligned with ⟨IN_AND_OUT-087-3⟩, although in neither case do the IO_PERSON slots point to the (correctly) aligned PERSON templates. Nonetheless, each pair yields two correct values, specifically for the NEW_STATUS and ON_THE_JOB slots.

This alignment is not possible with the restrictive algorithm, since it is performed solely on the basis of slots listed in Table 1. The restrictive algorithm finds the opposite alignment between the IN_AND_OUT templates, shown in Figure 8. This mapping also yields two correct slot fills for each IN_AND_OUT template, in this case, the IO_PERSON

COR		⟨IN_AND_OUT-087-1⟩	⟨IN_AND_OUT-087-1⟩
inc	IO_PERSON:	⟨PERSON-087-1⟩	⟨PERSON-087-8⟩
cor	NEW_STATUS:	OUT	OUT
cor	ON_THE_JOB:	UNCLEAR	UNCLEAR
COR		⟨IN_AND_OUT-087-4⟩	⟨IN_AND_OUT-087-3⟩
inc	IO_PERSON:	⟨PERSON-087-5⟩	⟨PERSON-087-20⟩
cor	NEW_STATUS:	OUT	OUT
cor	ON_THE_JOB:	NO	NO
spu	OTHER_ORG:		⟨ORGANIZATION-087-3⟩
spu	REL_OTHER_ORG:		OUTSIDE_ORG
COR		⟨SUCCESSION_EVENT-087-1⟩	⟨SUCCESSION_EVENT-087-2⟩
cor	SUCCESSION_ORG:	⟨ORGANIZATION-087-1⟩	⟨ORGANIZATION-087-9⟩
cor	POST:	"PRESIDENT"	"PRESIDENT"
cor	IN_AND_OUT:	⟨IN_AND_OUT-087-2⟩	⟨IN_AND_OUT-087-4⟩
inc		⟨IN_AND_OUT-087-1⟩	⟨IN_AND_OUT-087-3⟩
cor	VACANCY_REASON:	REASSIGNMENT	REASSIGNMENT
COR		⟨SUCCESSION_EVENT-087-2⟩	⟨SUCCESSION_EVENT-087-1⟩
inc	SUCCESSION_ORG:	⟨ORGANIZATION-087-1⟩	⟨ORGANIZATION-087-3⟩
cor	POST:	"CHIEF EXECUTIVE"	"CHIEF EXECUTIVE"
cor	IN_AND_OUT:	⟨IN_AND_OUT-087-3⟩	⟨IN_AND_OUT-087-2⟩
inc		⟨IN_AND_OUT-087-4⟩	⟨IN_AND_OUT-087-1⟩
inc	VACANCY_REASON:	REASSIGNMENT	OTH_UNK

Figure 7
Alignment for text 9403100087 with the standard algorithm.

COR		⟨IN_AND_OUT-087-4⟩	⟨IN_AND_OUT-087-1⟩
cor	IO_PERSON:	⟨PERSON-087-5⟩	⟨PERSON-087-8⟩
cor	NEW_STATUS:	OUT	OUT
inc	ON_THE_JOB:	NO	UNCLEAR
COR		⟨IN_AND_OUT-087-1⟩	⟨IN_AND_OUT-087-3⟩
cor	IO_PERSON:	⟨PERSON-087-1⟩	⟨PERSON-087-20⟩
cor	NEW_STATUS:	OUT	OUT
inc	ON_THE_JOB:	UNCLEAR	NO
spu	OTHER_ORG:		⟨ORGANIZATION-087-3⟩
spu	REL_OTHER_ORG:		OUTSIDE_ORG
COR		⟨SUCCESSION_EVENT-087-1⟩	⟨SUCCESSION_EVENT-087-2⟩
cor	SUCCESSION_ORG:	⟨ORGANIZATION-087-1⟩	⟨ORGANIZATION-087-9⟩
cor	POST:	"PRESIDENT"	"PRESIDENT"
cor	IN_AND_OUT:	⟨IN_AND_OUT-087-1⟩	⟨IN_AND_OUT-087-3⟩
cor		⟨IN_AND_OUT-087-2⟩	⟨IN_AND_OUT-087-4⟩
cor	VACANCY_REASON:	REASSIGNMENT	REASSIGNMENT
COR		⟨SUCCESSION_EVENT-087-2⟩	⟨SUCCESSION_EVENT-087-1⟩
inc	SUCCESSION_ORG:	⟨ORGANIZATION-087-1⟩	⟨ORGANIZATION-087-3⟩
cor	POST:	"CHIEF EXECUTIVE"	"CHIEF EXECUTIVE"
cor	IN_AND_OUT:	⟨IN_AND_OUT-087-4⟩	⟨IN_AND_OUT-087-1⟩
cor		⟨IN_AND_OUT-087-3⟩	⟨IN_AND_OUT-087-2⟩
inc	VACANCY_REASON:	REASSIGNMENT	OTH_UNK

Figure 8
Alignment for text 9403100087 with the restrictive algorithm.

and NEW_STATUS slots. This alignment—which is the correct one—results in an F-score of 75.00: Despite the fact that the SUCCESSION_EVENT templates are aligned the same way in both cases, the restrictive algorithm alignment leads to two additional correct fills for the pointers to the properly aligned IN_AND_OUT templates. While this alignment was a possible solution for the standard algorithm, it arbitrarily chose the wrong pairing of IN_AND_OUT templates—both possibilities resulted in two shared slot values—and thus the system was denied more than five points of F-score on the article.

This behavior was not specific to this example in our system runs; 18 of the 53 relevant texts (34%) displayed this behavior for at least one of the 53 system states. An average of 2.7 documents rose in score per system run. The magnitude of the increase in document score ranged from 0.25 to 7.27 points of F-score.

To sum, the more accurate alignment criterion affected approximately 24 out of 53 relevant documents (45%) for an average system run. In most cases, the effect was to reduce the score assigned by the standard algorithm, since the best-scoring (albeit incorrect) alignment found by the standard algorithm was often disallowed by the restrictive algorithm. However, due to the fact that the heuristic search process used for alignment is less likely to find the optimal mapping with the standard criterion, there were also cases in which the effect was to increase the score.

4.2.2 Diverging Indications Between System States. We now ask to what extent these documents exhibit the behaviors seen in the four development scenarios described in Section 3. In the first two of these scenarios, a system change that should have improved F-score decreased it instead. For these, we would expect the restrictive algorithm to correctly indicate an improvement. In the second two scenarios, a system change that should have decreased F-score increased it instead. For these, we would expect the restrictive algorithm to correctly indicate a reduction. Thus, we are interested in identifying those documents for which one algorithm signaled an improvement and the other signaled a reduction in score between a pair of system states.

It turns out that the output for 42 of the 53 relevant documents (79%) displayed this behavior for at least one pair of system states. The magnitude of the difference between document scores varied greatly, from 0.10 to 25.00 points of F-score. In the case of the 25.00 point difference, the standard algorithm indicated a change from 59.62 to 51.69, whereas the restrictive algorithm indicated a change from 34.62 to 51.69. Thus, while both algorithms assigned the same score to the second system state, the standard algorithm had assigned undeserved credit to the first system state, and what should have resulted in a 17-point improvement was shown instead as an 8-point reduction.

To sum, the problems we noted in Section 3 are not peculiar to the example shown in Figure 2; examples exhibiting this behavior are readily found in practice.

4.3 Effects of Stricter Alignment on Entire MUC-6 Development Set Results

In an actual development setting, of course, developers generally do not focus on differences in F-score for a single text, but rely instead on the scores assigned to the entire 100-text development corpus. Although the previous discussion showed that our development scenarios occur in practice, it is quite possible that these document-level differences were inconsequential in terms of the feedback obtained for the entire development set. We thus look at the difference between the algorithms with respect to the scores they assign to this larger set.

4.3.1 Effect on Scores. Unsurprisingly, the restrictive algorithm assigned a lower overall score than the standard algorithm for all 53 system states. The magnitude of the reduction ranged from 2.09 to 4.97 points of F-score, averaging 3.29 points.

4.3.2 Diverging Indications Between System States. We now ask if the two algorithms ever disagree about whether the difference between two system states constitutes a positive or negative change. This occurred for 90 of the 1,378 system state pairs (6.5%). The magnitude of the difference between the changes of performance measured by each algorithm ranged from 0.14 to 2.55 points, averaging 1.38. The magnitude was

less than 1 point in 33 cases, between 1 and 2 points in 42 cases, and was greater than 2 points in 15 cases.

These differences, and in particular those in the 2-point range, are large enough that they could affect a developer's decision about whether to adopt proposed system changes. In the case in which the difference was 2.55 points, for instance, a positive change was reported by the standard algorithm as a negative one: It indicated a decrement in performance of 1.26 points (from 56.74 to 55.48), whereas the restrictive algorithm indicated an increment of 1.29 points (from 51.84 to 53.13). In our experience, by the time an IE system is in the 50-point performance range on the MUC-6 task, the majority of further progress results from a series of incremental changes that have a relatively small affect on the overall score. Thus, a change that decreases performance by 1.26 points will almost certainly be removed from the system, whereas a change that increases performance by 1.29 will almost certainly be kept.

There were also cases in which a negative change was reported by the standard algorithm as a positive one. For one pair of system states, for example, the standard algorithm indicated an increase of 2.13 points (from 51.49 to 53.62), whereas the restrictive algorithm indicated a reduction of 0.26 (from 48.91 to 48.65). Again, this difference could cause a developer to keep rules in the system that negatively impact the quality of its output.

4.4 Summary

A study of the results of a 13 1/2-month effort focused on the MUC-6 task suggests that the problems described in Section 3 are not merely pedagogical, but can and do occur in actual practice. These problems are prevalent enough that they could realistically affect the technology development process in an adverse manner.

While it would be difficult to determine the extent to which these problems may have affected development in MUC-6, we take care to note that we do not find that their severity was so strong that they, in and of themselves, compromised the integrity of the MUC-6 evaluation process. Indeed, the existing evidence suggests that any impact the alignment criterion may have had on the manner in which MUC-6 systems were developed, as well as how they were ranked with respect to each other in the final evaluation, was not likely to have been dramatic.

The outcome of our study is therefore dually positive, in that the results demonstrate the *potential* of a lax alignment strategy to have a dramatically adverse effect on the technology development process, without this potential having actually been fully realized in MUC-6 itself. It should be borne in mind that it would not be difficult to construct a scenario in which the ramifications for a MUC-like task would have been far more severe—with a different task specification, template structure, set of slot definitions, or scoring scheme, for instance—to the extent that the integrity of such an evaluation could be compromised. Thus, these results serve to highlight a potential pitfall to be avoided in future evaluations of IE and other high-level language processing tasks, which, like the MUCs, may be the principle driving forces behind technology development for extended periods of time.

5. Conclusions

Methods for evaluating NLP systems are essential for tracking progress during the technology development process. To properly drive this process in both system building and machine learning settings, it is crucial that positive and negative modifications be reflected as such in the feedback provided by the scoring mechanism. We have presented several scenarios which demonstrate that the alignment strategy employed in

the MUC-6 evaluation creates a situation in which this requirement is not respected. Furthermore, we have shown that the problem is pervasive enough that it could realistically impact the development process using a larger set of development data. These results argue strongly for the use of strict and accurate alignment criteria in future natural language evaluations and for maintaining the independence of alignment criteria and the mechanisms used to calculate scores. This lesson is important because alignment problems will likely become exacerbated in future evaluations, as the natural language applications addressed become yet more complex.

Acknowledgments

This work was completed while the first author was at SRI International. We would like to thank Nancy Chinchor, Lynette Hirschman, Jerry Hobbs, David Israel, Andreas Stolcke, Beth Sundheim, Mabry Tyson, Ralph Weischedel, and two anonymous reviewers for helpful comments on earlier versions of the paper. All opinions expressed herein remain our own. This work would not have been possible without the contributions of the following people at SRI: Jerry Hobbs, David Israel, Megumi Kameyama, David Martin, Karen Myers, and Mabry Tyson.

References

Aberdeen, John, John Burger, David Day, Lynette Hirschman, Patricia Robinson, and Marc Vilain. 1995. MITRE: Description of the *Alembic* system used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 141–155, Columbia, MD, November. Morgan Kaufmann.

Appelt, Douglas E., Jerry R. Hobbs, John Bear, David Israel, Megumi Kameyama, Andy Kehler, David Martin, Karen Myers, and Mabry Tyson. 1995. SRI International FASTUS system MUC-6 test results and analysis. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 237–248, Columbia, MD, November. Morgan Kaufmann.

Chinchor, Nancy and Gary Dungca. 1995. Four scorers and seven years ago: The scoring method for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 33–38, Columbia, MD, November. Morgan Kaufmann.

CSL Special Issue. 1998. Special Issue on Evaluation in Language and Speech Technology. *Computer Speech and Language*, 12(4).

Grishman, Ralph and Beth Sundheim. 1995. Design of the MUC-6 evaluation. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 1–11, Columbia, MD, November. Morgan Kaufmann.

Hirschman, L. 1998. The evolution of evaluation: Lessons from the message understanding conferences. *Computer Speech and Language*, 12(4):281–305.

Sparck-Jones, Karen and Julia Rose Galliers. 1996. *Evaluating Natural Language Processing Systems: An Analysis and Review*. Lecture notes in computer science, 1083. Springer.

van Rijsbergen, C. J. 1979. *Information Retrieval*. Butterworths, London.