Text Filtering in MUC-3 and MUC-4

David D. Lewis
Center for Information and Language Studies
University of Chicago
Chicago, IL 60637
lewis@tira.uchicago.edu
and
Richard M. Tong
Advanced Decision Systems
(a division of Booz, Allen & Hamilton)
1500 Plymouth Street
Mountain View, CA 94706
rtong@ads.com

Introduction

One of the changes from the Third (MUC-3) to the Fourth (MUC-4) Message Understanding Conference was the emergence of text filtering as an explicit topic of discussion. In this paper we examine text filtering in MUC systems with three goals in mind. First, we clarify the difference between two uses of the term "text filtering" in the context of data extraction systems, and put these phenomena in the context of prior research on information retrieval (IR).

Secondly, we discuss the use of text filtering components in MUC-3 and MUC-4 systems, and present a preliminary scheme for classifying data extraction systems in terms of the features over which they do text filtering.

Finally, we examine the text filtering effectiveness of MUC-3 and MUC-4 systems, and introduce some approaches to the evaluation of text filtering systems which may be of interest themselves. Two questions of crucial interest are whether sites improved their system level text filtering effectiveness from MUC-3 to MUC-4, and what the effectiveness of MUC systems would be on real world data streams. Because of changes in both test set and system design since MUC-3 we were not able to address the first question. However, with respect to the second question, we present preliminary evidence suggesting that the text filtering precision of MUC systems declines with the generality of the data stream they process, i.e. the proportion of relevant documents. The ramifications of this for future research and for operational systems are discussed.

Text Filtering and Information Retrieval

The term *text filtering* (or *relevance filtering*) has been used in the context of data extraction systems to refer to two rather different things. First, any data extraction system can be evaluated on the degree to which it extracts data for all and only those documents which actually contain legitimate data. Implicitly or explicitly the system makes a decision, for each document, whether to extract some data or to extract no data. We can refer to the result of these decisions as *system level text filtering*. Therefore, the effectiveness of system level text filtering is a property that can be measured for any data extraction system.

Second, from an architectural standpoint, data extraction systems can include components that mark documents or parts of documents as nonrelevant, so that later stages of processing are not applied to them, or are applied differently to them. We call these *text filtering components* or *text filters*.

In both system level text filtering and component level text filtering, we have a computer program classifying texts into two categories: relevant and nonrelevant. Text filtering in data extraction systems is therefore closely related to a number of text classification tasks that information retrieval researchers have investigated. It is useful to view these tasks as lying along a continuum according to the degree of care that goes into defining classes of documents, and the duration of interest in those closes:

1. *Text retrieval*: Computer selection of a subset of a document database to display in whole or summary form to a user in response to a user request. A user's division of documents into relevant and nonrelevant may be of relatively short-term interest, and may indeed change during a retrieval session.

2. *Text routing*, *text filtering*, and *SDI (selective dissemination of information)*: These terms refer to a loose collection of text classification tasks such as managing personal electronic mail, distribution of information to the appropriate person in an organization, and so on. Categories are of longer term interest than in text retrieval, may be defined by someone other than the consumer of information, and may be part of an organized scheme.

3. *Text categorization:* Classification of documents with respect to a set of one or more pre-existing categories. The categories are usually of long term interest and part of an organized scheme (hierarchical, disjoint, etc.).

Text filtering in data extraction systems falls toward the text categorization end of this continuum, since the two categories of interest (templates/no templates) are of long term interest and are given considerable attention system builders. One difference between MUC text filtering and most IR categorization tasks is the complexity of the category definition. A relevance judgment in MUC-3 or MUC-4 required distinguishing between terrorist events and state sponsored violence on the one hand and very similar guerilla warfare events on the other. The exclusion of discredited, nonspecific, and nonrecent events further complicated the definition of relevance. In contrast, the distinctions made in classical text categorization tasks, such as automated controlled vocabulary indexing, tend to be broader and involve fewer conditions.

Another distinction is that the proportion of relevant documents in the MUC test sets is much higher than in typical text categorization applications. The consequences of this will be discussed when we examine the system level text filtering results from MUC-3 and MUC-4.

Approaches to Text Filtering in MUC System Design

MUC-3 and MUC-4 systems varied widely in the extent to which they made use of text filtering components. Some used none at all, others used several kinds. In this section we summarize the uses of text filtering found in these systems. We break down the uses of text filtering into three classes, according to whether the filtering takes place before, during or after parsing. Note that we us the term "parsing" rather loosely here to refer not just to syntactic analysis, but to all substantive linguistic analysis. Thus our categories are:

1. *Pre-parse:* Filtering is performed before significant linguistic analysis. Entire documents or parts of documents are flagged as nonrelevant based on the presence or absence of particular words or simple patterns of words.

2. *Intra-parse:* Some documents or parts of documents are classified as nonrelevant by examining partially produced syntactic or semantic representations before linguistic analysis is complete.

3. *Post-parse:* Filtering is based on examining the final output of linguistic analysis, or examining actual templates that are otherwise ready to be output. The decision made is usually to allow or disallow an entire document or an entire template, rather than a part of a document.[1]

[1]Note, that we chose not make a further distinction between "post-analysis" and "post-templates" filtering even though we have examples of both in the MUC-3/MUC-4 systems. From a text filtering perspective, this distinction is not so interesting since they are both examples of filtering on complex semantic structures using domain specific heuristics.

Table 1: Summary of Text Filtering Functions for MUC3 and MUC4 Sites

System	Pre-parse	Intra-parse	Post-parse
	MUC-3/4	MUC-3/4	MUC-3/4
ADS	yes/***	—/***	no/***
BBN	no/no	yes/yes	no/yes
ConQuest	no/no	no/no	yes/no
GE	yes/yes	no/no	yes/yes
GE-CMU	***/yes	***/no	***/yes
GTE	no/***	no/***	yes/***
HUGHES	no/no	—/—	no/no
ITP	yes/***	no/***	yes/***
LSI	yes/yes	no/no	yes/yes
MDC	no/yes	no/no	yes/yes
MITRE	***/no	***/no	***/no
NMSU/Brandeis	***/yes	***/yes	***/no
NYU	yes/yes	no/no	yes/yes
PARAMAX	yes/yes	—/—	no/no
PRC	no/no	no/no	yes/yes
SRA	***/yes	***/no	***/no
SRI	yes/yes	no/no	yes/yes
UMASS	no/no	yes/yes	yes/yes
UMICH	***/yes	***/no	***/yes
UNL/USL	yes/***	—/***	no/***
USC	***/no	***/no	***/no

LEGEND:
— denotes "not applicable"
*** denotes "did not participate"

The lines between these classes are not sharp. Filtering rules which refer to semantic classes of words, or which are based on complicated patterns defined over words, straddle the pre-parse and intra-parse cases. Similarly, the point at which parsing is over and any filtering would be post-parse is not always clear. Nevertheless, we found the distinctions useful, and summarize our analysis of the systems with respect to them in Table 1. The entries in this table are primarily derived from the system descriptions provided by the individual sites in the MUC-3 proceedings in their presentations at the MUC-4 workshop. In some cases, clarification was provided by sites in response to requests by the authors.

Use of Text Filtering in MUC-3 and MUC-4 Systems

In this section we briefly describe the text filtering processes performed by each of the sites participating in MUC-3 and/or MUC-4. While every site that participated in both MUC-3 and MUC-4 made some changes to their overall system these were mostly not in the text filtering functions. On the other hand, some repeating sites fielded completely different systems in MUC-4.[2]

ADS: ADS' CODEX system used a pre-parse filter based on matching complex patterns of words and phrases at the sentence level. Sentences that matched the patterns above some threshold were candidates for further processing. Those that were below threshold were eliminated. ADS participated in MUC-3 but not in MUC-4.

BBN: BBN's PLUM system performed intra-parse filtering in a "Discourse Module" where only

[2]More detailed descriptions of the MUC-4 systems can be found elsewhere in this volume. Detailed descriptions of the MUC-3 systems can be found in "Proceedings Third Message Understanding Conference (MUC-3)," San Diego, May 1991. Morgan Kaufmann Publishers, San Mateo, CA.

terrorist incidents, or possible terrorist incidents, generate "discourse event structures." The version of PLUM fielded at MUC-4 added a statistical classification algorithm for paragraph level filtering. This probabilistic filter was trained on paragraph level relevance judgments and flagged paragraphs as relevant or nonrelevant based on the presence of particular word stems. Full NL analysis of nonrelevant paragraphs was still carried out, but events in those paragraphs were not allowed to trigger templates. Thus this method mixed aspects of pre-parse and post-parse filtering. (We classify it as post-parse in Table 1.)

GE: GE's NLToolset performed pre-parse filtering of non-relevant sentences and parts of sentences using lexico-semantic patterns. Post-parse filtering involved generating templates in their final form and then checking them against relevance conditions.

GE-CMU: The GE-CMU system fielded for MUC-4 was identical to the GE system except for the replacement of GE's TRUMP analyzer with a generalized LR parser. Filtering functions are exactly the same. GE-CMU were new participants in MUC-4.

GTE: GTE's TIA system performed post-parse filtering on the semantic output of their semantic analyzer. This "output translator" was only partially implemented, however. GTE did not participate in MUC-4.

HUGHES: Hughes' approach made substantial use of text classification methods, but apparently did not use classifiers trained specifically on the distinction between relevant and nonrelevant texts.

ITP: ITP's system used a simple form of pre-parse filtering by ignoring sentences which did not contain a terrorism word. The system also performed post-parse filtering on the output of its language analysis module. The "cognitive model" produced by this module is an interpretation of the complete message text which was then examined for indicators of relevant MUC events. ITP did not participate in MUC-4.

LSI: LSI's DBG system performed both pre-parse and post-parse filtering. The pre-parse filtering was done by selecting only those sentences that contained a keyword taken from either the "event word list" or the "result word list." Post-parse filtering was based on processing semantic interpretations which, if they contained relevance indicators, caused the generation of an entry on the internal DBG-template-queue. This queue was subjected to further processing to determine which MUC templates should be generated.

MDC: MDC's INLET system fielded for MUC-3 used a preliminary form of post-parse filtering in the generation of templates. For MUC-4, MDC fielded an improved system called TexUS that performed keyword-pattern-based, pre-parse filtering at the sentence level, as well as some post-parse filtering based on event rejection while attempting to fill templates.

MITRE: Mitre's system performed no filtering as defined in this paper. Mitre was a new participant at MUC-4.

NMSU/Brandeis: The NMSU/Brandeis MucBruce system performed "two-pass" pre-parse filtering to determine both relevant texts and then relevant paragraphs with respect to specific events. These filters are statistical classifiers developed using the MUC training corpus. Intra-parse filtering was performed as part of the "robust parsing" process. Sentence level parsing was terminated if either no appropriate action word was found or various template filling criteria were not met. The NMSU/Brandeis team were new participants at MUC-4.

NYU: NYU's PROTEUS system performed pre-parse filtering at the sentence level using simple keyword pattern matching. This filtering is done after lexical analysis to minimize the number of patterns that need to be specified. Post-parse filtering was performed on the "event frames" generated from the output of the linguistic analysis. In particular, filters removed frames involving only military targets and those involving events more than two months old.

PARAMAX (MUC-3: UNISYS): Paramax's CBAS system fielded at MUC-4 was a more efficient implementation of the KBIRD rule base used in MUC-3. The CBAS system performed pre-parse filtering by using statistical correlations between words occurring in the text and "concept profiles" for MUC related events. Text was marked as relevant if the match was above a specific threshold. Subsequent processing keyed off the text marked as relevant.

PRC: PRC's PAKTUS system performed post-parse filtering by examining the output of the "discourse analysis" module. The filter looked for domain-specific patterns in the output and if no patterns were matched within a sentence, then that sentence was ignored in generating the final template output.

SRA: SRA's SOLOMON system performed pre-parse filtering by rejecting paragraphs that did not contain MUC-specific keywords. SRA were new participants in MUC-4.

SRI: SRI's TACITUS system fielded for MUC-3 used both pre-parse and post-parse filtering. The pre-parse filter was a two-stage function. First, it identified relevant sentences using a statistical n-gram model trained on the MUC corpus. This was followed by an "keyword antifilter" that flagged as relevant sentences that had been filtered out by the previous stage, but which contained certain keywords and were in close proximity to a sentence already marked as relevant. The post-parse filter operated on completed templates and removed those that did not satisfy certain coherence or relevance criteria. The completely new FASTUS system fielded for MUC-4 again used both pre-parse and post-parse filtering. Pre-parse filtering used "trigger words" derived from patterns recognized by the system as it constructed MUC templates. A variety of optional post-parse filters were used, such as screening out templates with too many targets, templates with civilian targets, and templates with stale dates.

Univ. of Maryland/ConQuest (MUC-3: Synchronetics): The system fielded for MUC-3 had a rudimentary post-parse filter that examined the semantic network output of the linguistic analysis component and determined whether any of the actions detected were within the parameters of a "reportable action" as defined by the MUC task. The ICOTAN system fielded for MUC-4 performed no text filtering as defined in this paper.

Univ. of Massachusetts: The University of Massachusetts system performed both intra-parse and post-parse filtering. The intra-parse filtering was achieved by screening input sentences for triggers associated with "concept nodes". If the input contained no triggers then no case frames were instantiated and no further processing of that input was performed. Post-parse filtering was done using a memory-based consolidation of case frames into event descriptions. Event descriptions that did not meet MUC specific criteria were not used to generate filled templates.

Univ. of Michigan: The University of Michigan's LINK system performed pre-parse filtering at the sentence level. A sentence was filtered if it contains no "interesting" definitions. A definition was interesting if it was one of the terrorist acts or related words. Post-parse filtering was a two-step process. First, inappropriate slot fillers were removed, then entire templates were filtered if they contained an inappropriate date, or if they did not contain enough fillers to enable a possible match with the answer key. The University of Michigan were new participants at MUC-4.

Univ. of Nebraska Lincoln/Univ. of Southwestern Louisiana: The UNL/USL system was based on statistical text categorization. Templates were created for a document if it passed through the "optimal query" filter (trained to distinguish between relevant and nonrelevant documents) or if it produced sufficient activation of some incident type and other associated concepts. UNL/USL did not participate in MUC-4.

Univ. of Southern California: USC's SNAP systems performed no text filtering in the sense used in this paper. USC were new participants in MUC-4.

Discussion

The most notable feature of Table 1 is that nearly all the systems include some filtering component. The relative rarity of intra-parse filtering is perhaps surprising, though without detailed descriptions of the parsing process, we admittedly were often unable to decide whether filtering was taking place. In any case, filtering during linguistic analysis is often an implicit process—the fact that an analysis did not produce a recognizable interpretation of a text unit and, therefore, ignored that unit is kind of filtering, but is not the kind under discussion here.

There was a tendency for certain constraints to be implemented as filters at particular points in the analysis process, and by particular methods. A broad distinction between potentially relevant text involving violent events, and nonrelevant text lacking such events, was often encoded as a pre-parse filter. These pre-parse filters usually checked text units for words strongly associated with one or more incident types, and often were trained by statistical methods. This seems appropriate, since statistical classification methods using words as features are probably at their best in making such broad distinctions in subject matter.

On the other hand, constraints such those against events which were too old, too vague, or which involved military targets typically were implemented as simple rules which examined either the output of semantic interpretation or, more commonly, templates generated in otherwise final form. Again, this seems a reasonable strategy since the constraints on relevance here referred to properties of data which should be observable in correctly generated templates.

As well as exhibiting a range of text filtering strategies, MUC sites also exhibited a range of motivations for their use of text filtering. One motivation was efficiency, particularly in MUC-3, where many sites were challenged by a test set size larger than that used in previous Message Understanding Conferences, or in traditional NLP research. The use of computationally cheap filtering components to limit the amount of text analyzed by expensive natural language methods was crucial to allowing several systems to process the test set.

In other cases, filtering components were intended to improve effectiveness rather than, or in addition to, efficiency. Filtering out or flagging nonrelevant text early in processing reduced the chance of later stages of analysis extracting spurious data. Alternately, plausible data was tentatively extracted from parts of a document, but a later filtering stage could reject the as a whole if it did not satisfy the relevance constraints.

A third motivation for including filtering components was to save effort by system builders. Filtering is a simple classification process, which means that statistical and machine learning techniques can be used to induce filters from training data with relatively little human effort. MUC-3 and MUC-4 participants used corpora tagged for relevance of messages, paragraphs, or even sentences in training filters. These labelled corpora have the advantage that they can be shared between sites more easily than actual code, and can support the investigation of a variety of filtering strategies.

Finally, many filtering techniques include thresholds that can be varied to make a text more or less apt to be filtered out, providing a degree of flexibility to the overall system. Varying the threshold on the text filter can be used directly to vary the recall and precision of system level text filtering, as well as indirectly to vary recall and precision of data extraction.

Unfortunately, it is difficult to tell how useful text filtering components were in achieving the above goals. As we write this article, we know of only two MUC-4 sites that have presented data on the effect of running their system with different filtering variations. BBN showed the ability to trade off data extraction recall and precision by varying the threshold on their paragraph level text filter. SRI investigated turning on or off four template-level post-parse filters. Two were found to improve both recall and precision (for overall data extraction), while two were found to produce recall/precision tradeoffs.

We hope there will be more experiments of this kind, even (or especially!) if they produce seemingly trivial results, such as performance being terrible without the filter, or performance being unchanged without the filter.

Evaluation of Text Filtering in MUC

	Yes is Correct	No is Correct	
Decides Yes	$a+x$	b	$a+b+x$
Decides No	c	$d+y$	$c+d+y$
	$a+c+x$	$b+d+y$	$a+b+c+d+x+y=n$

Figure 1: Contingency table for binary decisions, including $x+y=o$ optional/don't care decisions.

The MUC workshops have borrowed several of their effectiveness measures from information retrieval. As used in evaluating data extraction, the measures had to be modified considerably, so that the MUC measures recall, precision, and fallout are related to those in IR by analogy, not identity [CHL92].

However, when evaluating the degree to which MUC systems select all and only relevant documents from which to extract data, the measures recall, precision, and fallout have essentially the same meaning they do in text retrieval or text categorization. In all cases, we are evaluating the ability of a system to make correct yes/no decisions, and the contingency table model from signal detection is appropriate [Swe69].

One difference from typical IR tasks is the presence in MUC of documents for which all templates are optional, i.e. messages for which either decision by a system (generate templates or don't generate templates) is treated as correct. In Figure 1 we give a contingency table for binary decisions, adapted to allow for $x+y=o$ optional documents, as well as $a+c$ relevant documents and $b+d$ nonrelevant documents. The total in the yes/yes cell is the number of relevant documents that were judged relevant by the system (a), plus the number of optional documents that were judged relevant by the system (x). Similarly, the total in the no/no cell is the sum of the nonrelevant documents that were judged nonrelevant (d), and the optional documents that were judged nonrelevant (y). The effectiveness measures for system level text filtering used in MUC-4 were:

(1) recall $= (a+x)/(a+c+x)$

(2) precision $= (a+x)/(a+b+x)$

(3) fallout $= b/(b+d+y)$

When x and y are 0 (no optional documents) these reduce to the conventional measures used in IR.

Another measure used in IR is *generality*, or the proportion of documents in the test set which are relevant [Rob69]. It is not a measure of system effectiveness, but rather of a property of a test set. In the traditional IR case, when there are no optional documents, generality is defined as:

(4) generality $= (a+c)/(a+b+c+d)$

In the presence of optional documents, we would again like a definition of generality which is a measurement of the properties of the test set, not of any particular system. (Thus we should not refer to x or y in the formula, only o.) Two alternatives suggest themselves:

(4$'$) generality $= (a+c)/(a+b+c+d+o)$

(4$''$) generality $= (a+c+o)/(a+b+c+d+o)$

	Yes is Correct	No is Correct	
Decides Yes	$(r+o)s$	ns	$(r+n+o)s$
Decides No	$r(1-s)$	$(n+o)(1-s)$	$(r+n+o)(1-s)$
	$r+os$	$n+o(1-s)$	$r+n+o=t$

Figure 2: Expected values of contingency table cell counts for a system which randomly guesses a proportion s of documents to be relevant.

From a mathematical standpoint, the two meansures seem equally reasonable. We will use 4″ in this paper for reasons described below.

We mentioned in the previous section that the proportion of relevant documents is much higher in MUC than in most IR tasks. This means that chance effectiveness, i.e. the effectiveness we would expect a system to achieve by random guessing, is relatively high. Along with the presence of optional messages, this means that intuitions about whether particular effectiveness values constitute good performance can be misleading.

One aid to interpreting data is significance testing [Chi92b]. Another aid is to present for comparison the effectiveness that would be achieved by a system that performed the task randomly. We can compute the expected values of the effectiveness measures for a randomly performing text filter as follows. Suppose there are r relevant documents, n nonrelevant documents, and o optional documents in the test set. Assume a hypothetical system which will guess randomly with probability s that any particular document is relevant, and with probability $1-s$ will consider it nonrelevant. This gives us the contingency table shown in Figure 2. We show in each cell the expected number of documents which would fall in that cell, under the assumption that the system is given the benefit of the doubt for optional documents, as in MUC.

From this table we can calculate the expected values for recall, precision, and fallout:

(5) $E(\text{recall}) = \frac{(r+o)s}{r+os}$

(6) $E(\text{precision}) = \frac{r+o}{r+n+o}$

(7) $E(\text{fallout}) = \frac{ns}{n+o(1-s)}$

For example, the TST3 test set had 65 relevant, 4 optional, and 31 nonrelevant documents. A system which randomly treated 70% of documents as relevant would have an expected recall on TST3 of 0.71, expected precision of .69, and expected fallout of .67. A system that randomly treated 25% of documents as relevant would have expected recall of .26, expected precision of .69, and expected fallout of .23.

Note that the expected value of precision for a randomly guessing system is constant with varying values of s. In fact, it is simply generality, as defined in 4″ above. (Defining generality as in 4″ lets us maintain the identity between generality and the expected value of precision, even when optional documents are present.)

This points up a weakness in using precision to measure text filtering effectiveness. When systems are operating at chance or near chance levels, precision cannot distinguish between a tendency to blindly guess many documents as relevant and a tendency to blindly guess few documents as relevant. One consequence of this is that the best strategy for a system with poor text filtering capability, if evaluated on recall and precision, is to guess that every document is relevant, maximizing recall and ignoring precision.

Table 2: Recall, precision, and fallout for system level text filtering for MUC-3 and MUC-4 systems, plus a system which judges every document relevant. Values are in hundredths, i.e. 79 means 0.79.

	MUC-3 TST2 5/91			MUC-4 TST3 5/92			MUC-4 TST4 5/92		
Site	REC	PRE	FALL	REC	PRE	FALL	REC	PRE	FALL
ADS	91	77	51	***	***	***	***	***	***
BBN	100	69	85	83	87	23	83	74	31
CONQUEST	37	78	15	32	81	14	48	74	19
GE	95	81	38	91	87	28	96	84	22
GE-CMU	***	***	***	87	89	22	87	87	15
GTE	51	71	35	***	***	***	***	***	***
HUGHES	98	66	100	98	69	100	100	57	95
ITP	65	74	37	***	***	***	***	***	***
LSI	58	83	17	91	75	64	89	63	61
MDC	82	75	46	69	81	34	68	79	22
MITRE	***	***	***	90	76	59	100	64	70
NMSU	***	***	***	83	80	41	95	66	61
NYU	95	72	74	82	86	27	91	85	20
PARAMAX	60	90	10	88	77	56	96	68	56
PRC	98	78	53	73	86	24	72	76	26
SRA	***	***	***	71	85	23	91	75	35
SRI	70	83	24	90	88	25	93	82	28
UMASS	97	82	39	91	94	12	91	82	24
UMICH	***	***	***	90	82	41	82	69	44
UNL/USL	94	74	53	***	***	***	***	***	***
USC	***	***	***	45	70	39	69	70	31
AllRel	100	66	100	100	69	100	100	55	100

Fallout, on the other hand, correctly distinguishes systems which treat many documents as relevant from those that treat few documents as relevant, even if those decisions are nearly random. We will see other reasons to prefer fallout over precision as an effectiveness measure for text filtering.

It is important to remember, of course, that MUC-3 and MUC-4 system designers were attempting to achieve the best possible overall template filling effectiveness, not just the best text filtering effectiveness. The data extraction effectiveness of a system that randomly generated templates, not just relevance decisions, would of course be much worse than that of any actual MUC system.

Text Filtering Effectiveness in MUC-3 and MUC-4

In this section we present data on the effectiveness of system level text filtering in MUC-3 and MUC-4 systems. Table 2 presents the recall, precision, and fallout scores for text filtering for each MUC-3 site on the TST2 testset, and for each MUC-4 site on the TST3 and TST4 testsets. These scores were computed from the official MUC-3 and MUC-4 score reports. Along with the system scores, we also present a final row, *AllRel*, which indicates what scores a system would get if it simply treated every document as relevant.

Table 3 similarly presents the values of the F-measure [van79] for all MUC-3 and MUC-4 systems. We compute the F-measure for system level text filtering the same way as for overall system effectiveness [Chi92], but use the text filtering recall and precision values as input, rather than the data extraction recall and precision values:

$$F = \frac{(\beta^2 + 1.0) \times P \times R}{\beta^2 \times P + R}$$

Finally, Figures 3 to 8 plot recall vs. precision, and fallout vs. recall for each of the three data

Table 3: F-values for system level text filtering effectiveness of MUC-3 and MUC-4 sites, plus a hypothetical system (AllRel) which judges every document relevant. Values of F for β of 0.5, 1.0, and 2.0 are given. Values are in hundredths, i.e. 79 means 0.79.

	MUC-3 TST2 5/91			MUC-4 TST3 5/92			MUC-4 TST4 5/92		
Site	β 0.5	β 1.0	β 2.0	β 0.5	β 1.0	β 2.0	β 0.5	β 1.0	β 2.0
ADS	79	83	88	***	***	***	***	***	***
BBN	74	82	92	86	85	84	76	78	81
CONQUEST	64	50	41	62	46	36	67	58	52
GE	83	87	92	88	89	90	86	90	93
GE-CMU	***	***	***	89	88	87	87	87	87
GTE	66	59	54	***	***	***	***	***	***
HUGHES	71	79	89	73	81	90	62	73	87
ITP	72	69	67	***	***	***	***	***	***
LSI	76	68	62	78	82	87	67	74	82
MDC	76	78	80	78	75	71	77	73	70
MITRE	***	***	***	78	82	87	69	78	90
NMSU	***	***	***	81	81	82	70	78	87
NYU	76	82	89	85	84	83	86	88	90
PARAMAX	82	72	64	79	82	86	72	80	89
PRC	81	87	93	83	79	75	75	74	73
SRA	***	***	***	82	77	73	78	82	87
SRI	80	76	72	88	89	90	84	87	91
UMASS	85	89	94	93	92	92	84	86	89
UMICH	***	***	***	83	86	88	71	75	79
UNL/USL	77	83	89	***	***	***	***	***	***
USC	***	***	***	63	55	48	70	69	69
AllRel	71	80	91	74	82	92	60	71	86

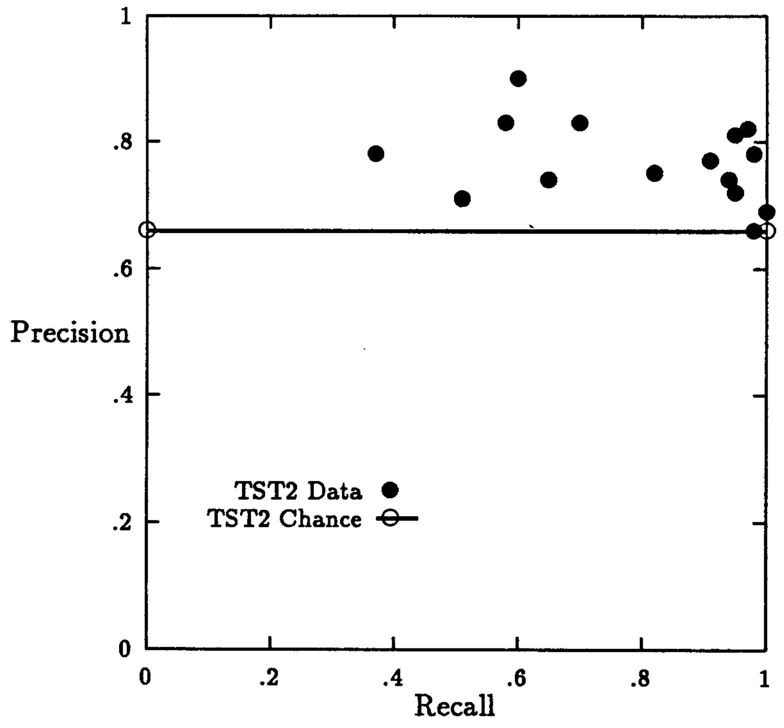

Figure 3: System level text filtering: recall vs. precision of MUC-3 systems on TST2.

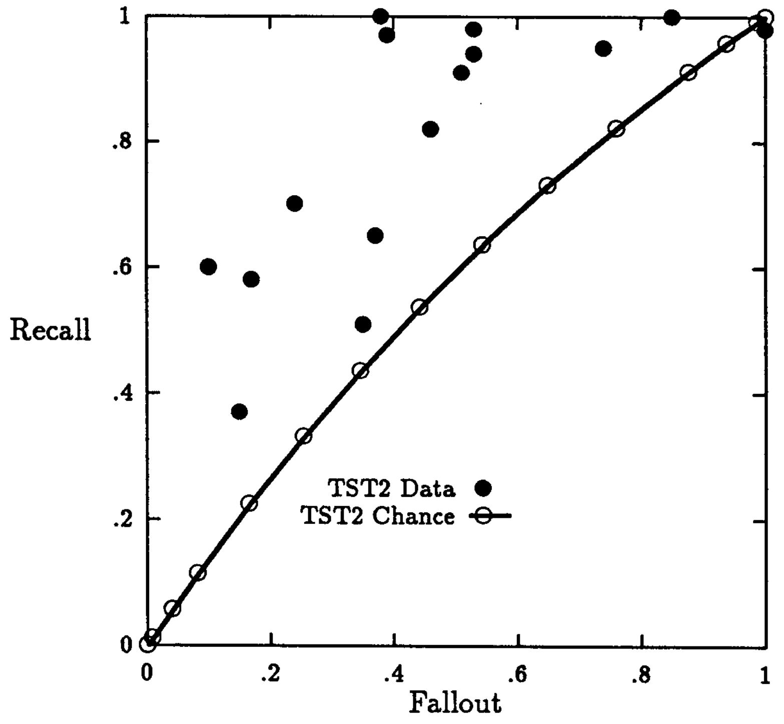

Figure 4: System level text filtering: fallout vs. recall (ROC plot) of MUC-3 systems on TST2.

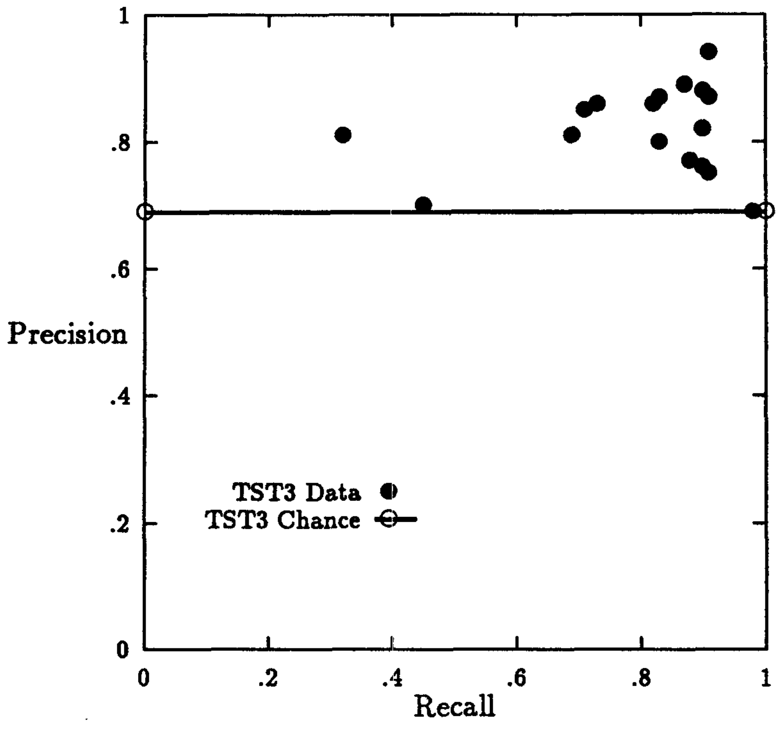

Figure 5: System level text filtering: recall vs. precision of MUC-4 systems on TST3.

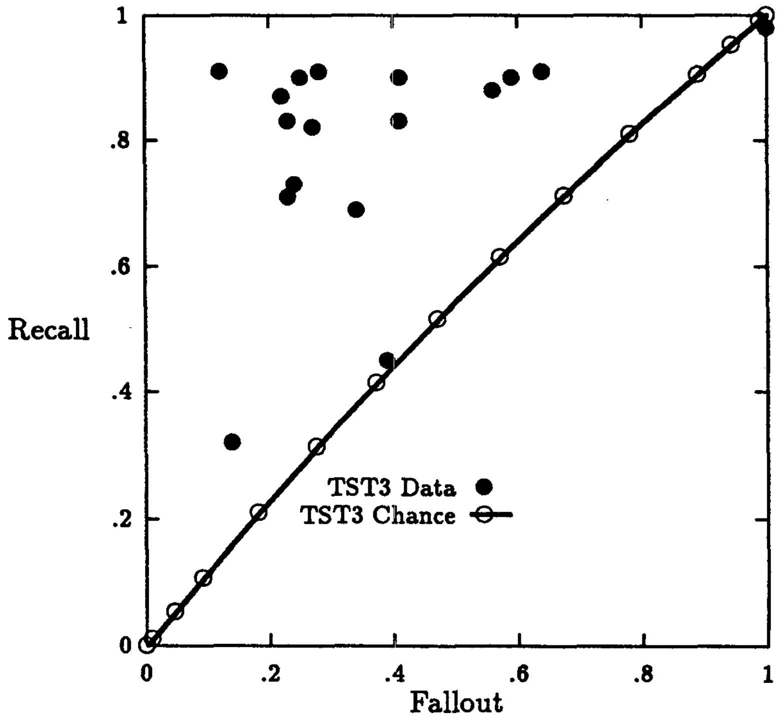

Figure 6: System level text filtering: fallout vs. recall (ROC plot) of MUC-4 systems on TST3.

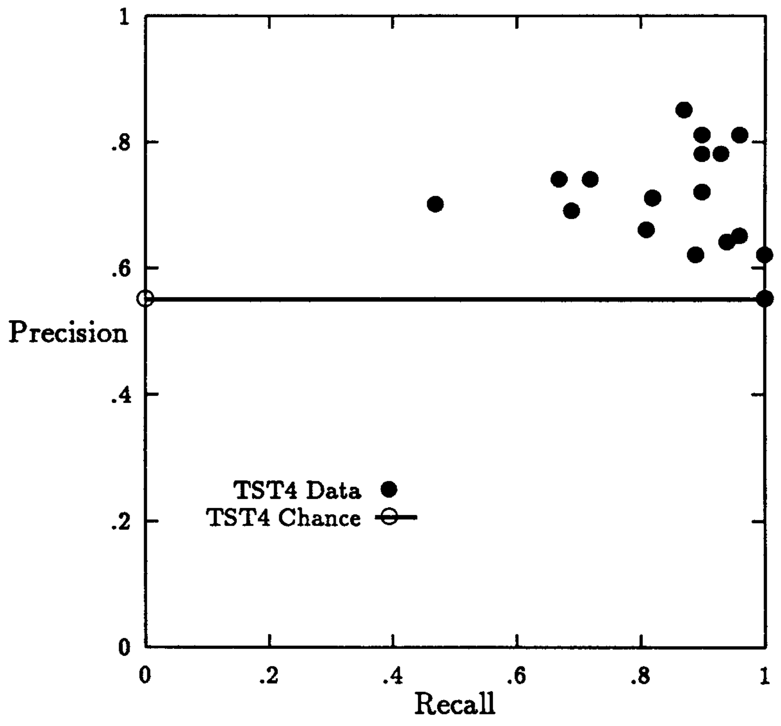

Figure 7: System level text filtering: recall vs. precision of MUC-4 systems on TST4.

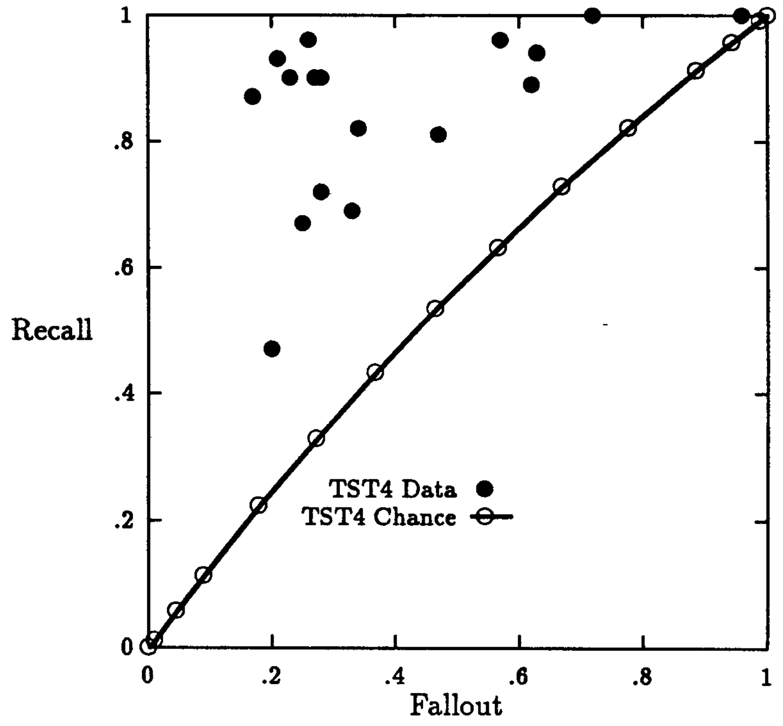

Figure 8: System level text filtering: fallout vs. recall (ROC plot) of MUC-4 systems on TST4.

sets. Displays of fallout vs. recall are often called ROC (relative operating characteristic) plots and are widely used in signal detection and decision theory. Along with the observed values for the MUC-3 and MUC-4 systems, we also plot the values that would be produced by a system which randomly classified some fraction s of documents as relevant. The graphs show how different values of s lead to different values of recall and fallout in such a system, while precision remains constant.

Analysis

The first thing that one notices about the system level text filtering results is the high level of effectiveness in absolute terms. For instance, Table 3 shows that the text filtering F-measure scores on TST3 range from .46 to .92 with a mean of .80 and a median of .82. On the other hand, as the AllRel row shows, the F-measure for a system that simply guessed every story to be relevant would also be .82. The MUC-4 systems stack up somewhat better on TST4, but still not startlingly better than a system that did no filtering at all.

We would like to know how much the text filtering ability of systems progressed from MUC-3 to MUC-4. Unfortunately, both the testsets and the systems themselves changed between MUC-3 and MUC-4. The TST2 and TST3 test sets were drawn from the same population of stories, but not randomly, and we also do not have data on the variability between 100 message samples from this population. Thus, while 8 of the 11 sites who participated in both MUC-3 and MUC-4 increased in their text filtering F-measure score (if we give equal weight to recall and precision, i.e. $\beta = 1.0$), we do not know whether this was a result of improvements in system design, the change of test set, random fluctuation, or some combination of these.

An issue of great interest is how robust MUC systems are to changes in the nature of their input text. Here the comparsion in effectiveness between TST3 and TST4 is informative, since exactly the same system configurations were run on the two data sets. TST4 was drawn from an earlier time slice of the same text stream as TST3 [Sun92], and so was a relatively mild test of portability to new texts. Of the 17 MUC-4 systems, 12 declined in the text filtering F-measure ($\beta = 1.0$) and 5 improved, from TST3 to TST4. However, the mean change in F-measure across the 12 systems which declined was only 4.9.[3] This compares favorably with the decline of 11.0 in F-measure for a strategy of guessing every document to be relevant.

However, going beyond the summary F-measure to the individual recall, precision, and fallout text filtering scores reveals a different picture. Examining Figures 5 and 7 shows that from TST3 to TST4 there was a substantial decline in precision, coinciding with the decline in generality of the testset or, equivalently, the expected precision of a randomly guessing system. In contrast, fallout remains relatively stable between TST3 and TST4, as seen in Figures 6 and 8.

We can quantify this by examining Table 2. For the 17 MUC-4 systems, the mean decline in precision from TST3 to TST4 was 7.9, while fallout increased by only 1.9. Taking a trimmed mean [Cha88] by discarding the highest and lowest differences as outliers makes the distinction even more apparent: precision drops by 8.0 but fallout increases by only 1.0.

The ramifications of this are two-fold. First, from the standpoint of choosing evaluation measures for system level text filtering, precision and, therefore, the F-measure, are handicapped by the strong correlation between precision and generality.[4] The correlation between precision and generality has been pointed out in the IR literature [Rob69], but is more serious in the MUC testsets. The generality for queries in a typical IR test collection is quite low. For example, generality for the 64 queries in the CACM-3204 test collection [FNL88] ranges from 0.000 to 0.016, putting a relatively low floor on precision. This contrasts with generalities for TST2, TST3, and TST4 of .66, .69, and .55 respectively, which constrain precision considerably.

Secondly, and more important, the generality of real world text streams will be much lower than that of the MUC test sets. The MUC data sets were a small subset of the original full text

[3] The mean change in F-measure for all 17 MUC-4 systems, including the five that increased in F-measure that increased was a decrease of only 1.4. However, that figure is heavily affected by two systems that did much better on TST4 than TST3, but nevertheless did poorly on both.

[4] It would, of course, be possible to define a single measure similar to the F-measure but based on recall and fallout, and this might be desirable.

databases of messages [Sun91]. First, only messages categorized as being about Latin America were used. Against that subset, a boolean query was used to find messages which contained a country name of interest as well as one of a set of 24 words highly suggestive of terrorist incidents were included in the MUC training and test sets. This further restriction resulted in a subset about 15% of the size of the original Latin American set [Sun92b].

The fact that only a small proportion of the database was used suggests that the raw text stream has a much lower generality than the MUC test sets. If MUC systems have constant fallout or, equivalently, precision which drops in proportion to generality, then system level text filtering effectiveness would be much lower on the original FBIS text stream than it was on the MUC subsets. Overall data extraction effectiveness is unlikely to be high if systems do poorly even at distinguishing from which documents to extract data.

Is the assumption of constant fallout too pessimistic? For one query-based text retrieval test collection, Salton found that both precision and fallout decreased by factors of 2 to 3 when generality was decreased by a factor of 7 [Sal72]. Thus precision decreased at a slower rate than would be the case if fallout was constant, but still at a substantial rate.

Even these figures are probably overoptimistic for the MUC case. Salton's experiment involved expanding the collection by a set of documents containing no relevant documents. The transition for the MUC testset to the raw FBIS data stream would involve adding new relevant as well as nonrelevant documents, and those new relevant documents would be harder to detect than the current ones (since they would not contain any of the 24 suggestive keywords, and would not have been classified by FBIS as being about Latin America). In order to obtain the same level of recall as on the MUC testsets, still more of a sacrifice in precision/fallout would likely need to be made.

Conclusions

In this article we introduced a distinction between system level text filtering, a characteristic which can be measured for any data extraction system, and component level text filtering, a strategy which is used by some data extraction systems. While we discussed the architectural choices that MUC-3 and MUC-4 systems have made with respect to component level text filtering, there was relatively little we could say about the ramifications of those choices, since few sites tested turning off or varying the nature of their text filtering components. We encourage more experiments of this sort. We also note that text filtering components, particularly those which operate on raw words or on completed templates rather than internal representations, are good candidates for interchangable parts in data extraction systems.

We were able to draw somewhat stronger conclusions about system level text filtering in MUC systems. The MUC-3 and MUC-4 systems exhibited a high absolute level of effectiveness at system level text filtering. We pointed out, however, that on the MUC test sets, a level of effectiveness equal to that of some systems could be achieved by blindly guessing which documents were relevant.

Of more consequence were the differences in text filtering effectiveness between test sets TST3 and TST4. While TST4 was chosen to provide a different time slice than TST3, it was more notable for having a lower generality (proportion of relevant documents) than TST3. The fact that precision declined on the average by 7.9 points from TST3 to TST4, while fallout increased by 1.0 points, raises the possibility that the effectiveness of MUC systems may drop dramatically if they are applied to real world text streams.

We also noted that, due to the correlation between precision and generality, fallout is a more useful effectiveness measure for understanding system behavior than is precision, even if operational requirements may sometimes more conveniently be stated in terms of precision.

Verifying whether this is the case will require testing of system level text filtering effectiveness on much larger databases, with a generality level similar to that of real world data streams. Fortunately, this is practical, since a large number of documents can be judged for relevance in the time it takes to create a data extraction answer key for a single document. Such an approach would also support the construction of better data extraction test sets, since a wide range of types of relevant documents could be sampled from, not just those which contain obvious keyword clues.

As a final note, we would like to stress the importance of measuring the degree of agreement among human coders both for the judging of relevance and the filling out of answer key templates. All analyses of the MUC-3 and MUC-4 data must be tempered by the fact that we do not know this degree of agreement, which research in IR has shown might be only 60% or lower [Cle91].

Acknowledgments

We thank DARPA for supporting the authors' travel to the MUC-4 workshop, and Beth Sundheim for her aid with this paper and her splendid handling of the MUC-4 process.

References

[Cha88] Chatfield, Christopher. *Problem Solving: A Statistician's Guide.* Chapman and Hall, London, 1988.

[Chi92] Chinchor, Nancy. MUC-4 evaluation metrics. In this volume.

[Chi92b] Chinchor, Nancy. The statistical significance of the MUC-4 results. In this volume.

[CHL92] Chinchor, Nancy; Hirschman, Lynette; and Lewis, David D. Evaluating message understanding systems: An analysis of the third message understanding conference (MUC-3). Submitted to *Computational Linguistics*, 1992.

[Cle91] Cleverdon, Cyril W. The significance of the Cranfield tests of index languages. In *Fourteenth International Conference on Research & Development in Information Retrieval*, pages 3–12, 1991.

[FNL88] Fox, Edward A., Nunn, Gary L., and Lee, Whay C. Coefficients for combining concept classes in a collection. In *Eleventh International Conference on Research & Development in Information Retrieval*, pages 291–307, 1988.

[Rob69] Robertson, S. E. The parametric description of retrieval tests. Part I: The basic parameters. *Journal of Documentation*, 25(1):1–27, 1969.

[Sal72] Salton, G. The "generality" effect and the retrieval evaluation for large collections. *Journal of the American Society for Information Science*, pages 11–22, January–February, 1972.

[Sun91] Sundheim, Beth M. Overview of the Third Message Understanding Evaluation and Conference. In *Proceedings of the Third Message Understanding Evaluation and Conference*, Los Altos, CA: Morgan Kaufmann, pages 3–16, May 1991.

[Sun92] Sundheim, Beth M. Overview of the Fourth Message Understanding Evaluation and Conference. In this volume.

[Sun92b] Sundheim, Beth M. Personal communication. August 13, 1992.

[Swe69] Swets, John A. Effectiveness of information retrieval methods. *American Documentation*, pages 72–89, January, 1969.

[van79] van Rijsbergen, C. J. *Information Retrieval.* Butterworths, London, second edition, 1979.