A Model-Based Analysis of Semi-Automated Data Discovery and Entry Using Automated Content Extraction

RANSOM WINDER and CRAIG HAIMSON
The MITRE Corporation, Annapolis Junction, MD
and
JADE GOLDSTEIN-STEWART and JUSTIN GROSSMAN
United States Department of Defense, Washington, DC

Content extraction systems can automatically extract entities and relations from raw text and use the information to populate knowledge bases, potentially eliminating the need for manual data discovery and entry. Unfortunately, content extraction is not sufficiently accurate for end-users who require high trust in the information uploaded to their databases, creating a need for human validation and correction of extracted content. In this paper, we examine content extraction errors and explore their influence on a prototype semi-automated system that allows a human reviewer to correct and validate extracted information before uploading it, focusing on the identification and correction of precision errors. We applied content extraction to six different corpora and used a Goals, Operators, Methods, and Selection rules Language (GOMSL) model to simulate the activities of a human using the prototype system to review extraction results, correct precision errors, ignore spurious instances, and validate information. We compared the simulated task completion rate of the semi-automated system model with that of a second GOMSL model that simulates the steps required for finding and entering information manually. Results quantify the efficiency advantage of the semi-automated workflow and illustrate the value of employing multidisciplinary quantitative methods to calculate system-level measures of technology utility.

Categories and Subject Descriptors: H.5.2 **[User Interfaces]**– Evaluation/methodology.
General Terms: Human Factors, Measurement, Performance
Additional Key Words and Phrases: Automated content extraction, data entry, GOMS, knowledge base, semi-automated workflow

1. INTRODUCTION

In the modern computer age, there is a rapidly growing quantity of digital documents containing primarily unstructured text with varying information quality. To fully exploit this information, it must be extracted and transferred to structured knowledge bases that support sophisticated data mining and exploration. Due to the superfluity of text in many domains, it is becoming infeasible for humans to perform this task manually; thus, automated content extraction systems [Moens 2006] are increasingly important tools for information discovery. Ideally, content extraction systems could be used to populate knowledge bases automatically.

However, to earn the trust of consumers, the information in a knowledge base must be highly accurate. If inaccurately extracted content is automatically uploaded to a database, the quality of information will suffer. How accurate is the current state-of-the-art extraction system? Evaluations conducted through programs such as the National Institute of Standards and Technology's Automated Content Extraction (ACE) initiative suggest there is still considerable room for improvement. Although there are reports of automatic entity extraction systems being trained to achieve precision scores of over 90% [Grishman and Sundheim 1996], there is evidence casting doubt that such systems are extensible even across similar corpora [Vilain et al. 2007]. Moreover, when entity extraction systems are required to coreference specific textual mentions of the same entity, their best precision scores only reach a range of 60-80% [Marsh and Perzanowski 1998], and performance on relation and event extraction tasks is even lower.

We conducted an extensive evaluation of extraction errors in our previous work [Goldstein-Stewart and Winder 2009] and concluded that content extraction should not be used to support knowledge base population without a human operator in the loop. We described a prototype semi-automated system (Fix Extractor Errors before Database Entry, or FEEDE) that allows a human reviewer to inspect, correct, and validate extracted information before it is translated to a structured form and uploaded. Operational users who have experimented with FEEDE have a favorable opinion of it. This kind of system is, to our knowledge, not well explored, though there have been past efforts on interfaces for data that come from partially formatted sources [Barclay et al. 1996] and extraction of biomolecular interaction information from journal article text [Donaldson et al. 2003] that also incorporated human review to eliminate errors introduced in automated extraction. Human inspection helps to ensure high quality of information in the database. Yet, since the human user acts as a bottleneck on system speed, the question remains: How much benefit can actually be achieved with methods that partner automatic extraction with human validation?

To answer this question, we conducted a model-based analysis of human-in-the-loop content extraction, building on an earlier study [Haimson and Grossman 2009] that used the Goals, Operators, Methods, and Selection rules (GOMS) paradigm [Card et al. 1983] to explore semi-automated content extraction/knowledge base population workflow. As its acronym suggests, GOMS breaks tasks down into goals (or a subgoal hierarchy) required for the task to be accomplished, operators that represent different activities of the task performer as a composite of perceptual, cognitive or motor functions, methods that appropriately sequence the operators and subgoal invocations, and selection rules that indicate which method is applied in a certain situation based on user knowledge [John and Kieras 1996]. GOMS estimates how long it would take a human user to complete the modeled task, using timing parameters derived from human performance experiments. It has been repeatedly demonstrated that GOMS can produce estimates that are within 10% or less of ground truth human performance data [John and Kieras 1996]. As with any model-based procedure, GOMS simulations can be used to conduct "what-if" experiments across a wide range of conditions with far less time, effort, and expense than would be required to conduct such exhaustive experimentation with human participants, or even a working prototype system. Moreover, GOMS can be used to simulate systems that are not available for testing (even systems that are not yet built).

We used GOMS to estimate how long it would take a human to review, correct, and submit information extracted from six different corpora. First, we applied content extraction technologies to each of these corpora and analyzed the results to determine what kinds of corrective actions a user would be required to perform in order to ready a result for submission to a knowledge base. We then used GOMS Language (GOMSL) [Kieras, 2006; Kieras et al. 1995] models to represent the corresponding workflows, which we simulated using the GOMSL Evaluation and Analysis (GLEAN) system. By comparing simulation results for the semi-automated workflows with those generated using a model of a fully manual information discovery and data entry workflow, we were able to quantify the expected efficiency advantage of using the semi-automated workflow to enter information extracted from the corpora. Below, we describe our approach, results, and implications concerning the viability of semi-automated content extraction technology.

2. WORKFLOW DESCRIPTIONS

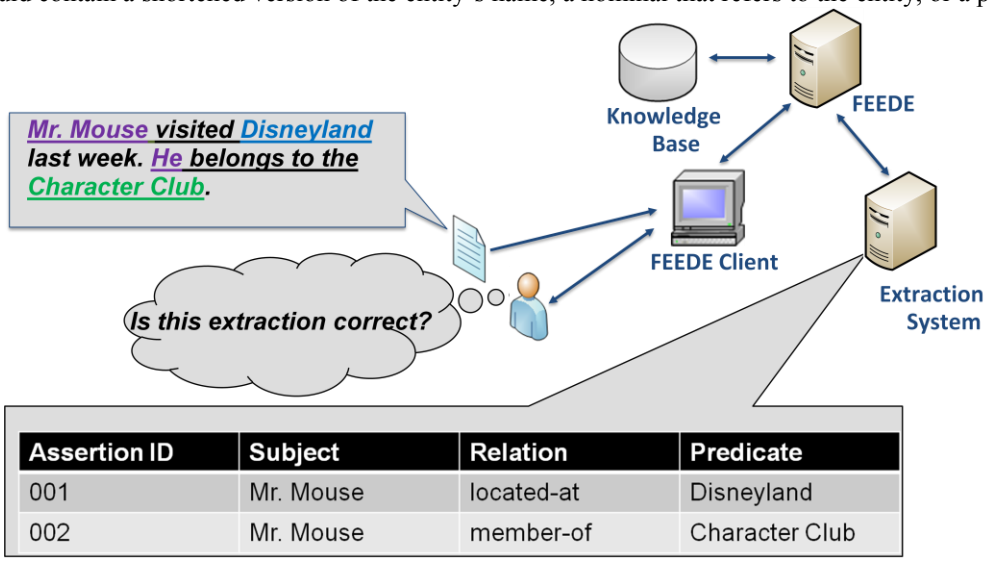
We begin by describing the semi-automated workflow and its manual counterpart (see our earlier paper [Goldstein-Stewart and Winder 2009] for lengthier discussion of the FEEDE system). We will refer to the semi-automated workflow as "SW" and the manual workflow as "MW". SW comprises the steps a user would actually perform to populate a knowledge base using FEEDE. In contrast, while MW is representative of real world tasks and systems, it is an idealized workflow designed to include the minimal number of human/system actions required to accomplish the same information extraction/knowledge base population task that FEEDE supports.

Note that both SW and MW are missing entity resolution steps. Entity resolution is the process of determining whether an entity about which one wishes to enter new information already exists in a database; if it does, the new information can be linked to the existing entity and combined with the information that has already been stored for that entity. Both SW and MW do require entity resolution. However, the entity resolution process is not affected by content extraction (users resolve entities in the same manner whether they discover them manually by reading or automatically by content extraction). Moreover, the decision making processes involved in disambiguating entities can be highly complex and beyond the scope of GOMSL [Kieras 2006]. Thus we chose to exclude entity resolution from our modeled workflows and instead focused on the information discovery and data entry processes.

SW starts when a user uploads a batch of raw text documents for automatic extraction of entities, relations, and document citations (it is assumed that documents contain structured headers with easily parsed reference information). In the ACE ontological framework ["ACE English" 2005], entities consist of textual "mentions" that refer to the same object (or set of objects), and relations consist of connections between entities, which can be represented as having four parts: the subject entity argument, the predicate or relation type, the object entity

argument, and a time argument, indicating when a relation held. Both entities and relations can be one of many different types and subtypes (e.g., an entity could be of type *person* and subtype *individual*).

A graphical user interface (GUI) presents extraction results as a set of "assertions", each of which includes a subject entity, a relation, and a predicate entity (Figure 1). The GUI hides duplicate assertions extracted from the same document; this prevents users from having to review redundant information, and we have found, in practice, that it results in only a very small percentage of unique assertions being lost due to their being incorrectly identified as a duplicate. The GUI initially presents users with a list of the subject entities that were extracted; by selecting a subject entity, the user can view all of the assertions extracted for that subject, and by selecting a given assertion, the user can view the document from which the assertion was extracted, with the text corresponding to subject entity, predicate entity, and relation highlighted (i.e., the extent of the relation and its immediate context). Note that the entities in an assertion are referred to by their primary entity name (PEN), which is assumed to be the longest named mention of that entity in the document; however, the actual span from which the corresponding relation was extracted could contain a shortened version of the entity's name, a nominal that refers to the entity, or a pronoun.



Assertion ID	Subject	Relation	Predicate
001	Mr. Mouse	located-at	Disneyland
002	Mr. Mouse	member-of	Character Club

Fig 1. Overview of SW

The user evaluates the assertion by comparing it with the highlighted material. If the source text and extraction do not match, the user must decide whether the assertion is correctable or spurious. If it can be corrected, the user edits the entity arguments, the entity types, or the relation type as necessary. Possible alternative entity and relation types are displayed within pulldown lists; to change a type, the user opens the corresponding list and selects the appropriate type. To change entity arguments, the user first consults a pulldown list of PENs that have been extracted from the same document. If the correct entity is absent from this list, the user can copy the name directly from the source document and paste it into the argument field. After editing the assertion as necessary, the user indicates that the assertion is valid. Once the user has evaluated, edited, and validated all of the assertions that were extracted, s/he can review and upload them to the knowledge base. For simplicity, we assume that SW users validate and submit all correct or correctable assertions to the database.

MW starts when a user opens a file containing a batch of documents and begins reading the documents, recording the document identification number, date, title, and classification for each in a dedicated field within a spreadsheet-like GUI. As s/he reads each document, s/he identifies the types of entities and relations that FEEDE extracts automatically for the SW user (Figure 2). The MW user determines whether s/he has already entered this information for the current document and, if s/he has not, she records the information in a spreadsheet client, within columns designated for the subject, relation (selected from a pulldown list of options), and object. For simplicity, we assume that MW users record all unique relations within each document and that an MW server mediates the transfer of data to the database.

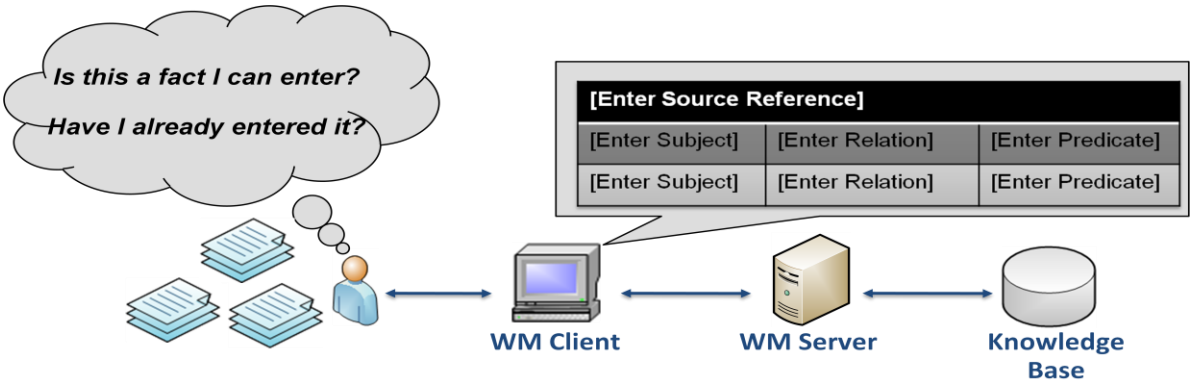


Fig 2. Overview of MW

Note that neither SW nor MW include an entity resolution process in which the user determines whether extracted/found entities already exist in the knowledge base. While an important component of the overall knowledge base population process, entity resolution does not leverage content extraction. For either workflow, entity resolution would involve querying the knowledge base with the extracted/found entity name and reviewing results to determine whether or not a returned entity matches the entity referenced in the source text. Moreover, making this determination relies on cognitive skills that are outside the scope of GOMSL [Kieras 2006]. Thus, we opted to focus our evaluation on aspects of the two workflows that expose the costs and benefits of using content extraction and that are amenable to GOMS analysis.

3. EXTRACTION ANALYSIS

3.1 Evaluation Method and Basic Results

To evaluate the impact of content extraction accuracy on the SW workflow, we needed a test corpus and content extraction technologies to apply to it. Content extraction systems search for patterns in raw text, and these patterns can be either hard-coded as rules or acquired via statistical learning techniques. To ensure a comprehensive assessment of content extraction performance, we selected both a leading statistical extraction technology (which we will refer to as ST) and also a leading rule-based extraction technology (which we will refer to as RB) as the systems on which to base our analysis of SW. For our test corpus, we obtained a set of 152 documents that had been manually annotated in accordance with ACE guidelines. The corpus contained examples from six distinct genres: broadcast conversation transcripts (BC – 9 documents), broadcast news transcripts (BN – 74 documents), telephone conversation transcripts (CTS – 6 documents), newswire (NW – 34 documents), usenet articles (UN – 12 documents), and weblogs (WL – 17 documents).

We submitted the corpus of documents to ST and RB and scored their performance using the ACE 2005 evaluation script (ace05-eval-v14a.pl). The evaluation script evaluates how closely ACE annotations generated by automatic content extraction match annotations humans have applied to the same document. An annotated document is referred to as "reference" when it is tagged by a human and as "hypothesis" when it is tagged by automatic content extractors. The scorer can produce both unweighted results (the default) and value-based results, which are derived by differentially weighting different types of errors in attributes such as type (e.g., person vs. organization) and roles (e.g., subject vs. object), as well as false alarms in the recognition of entities, relations, and arguments to those relations [ACE 2005]. There is an implicit assumption that reference documents, which are taken as ground truth, will be free from the sorts of errors encountered in hypothesis. This is somewhat idealized when one considers there is an imperfect inter-annotator agreement for entities and relations in ACE. However, for the purposes of this study, we consider matching human performance to be the standard to achieve and assume we are testing the upper boundary of the human performance.

Mentions of entities and relations are defined by specific offsets in the annotated text. The scorer identifies potential correspondences between entity mentions in reference and hypothesis documents when there is at least a 30% overlap between their character sequences. Note that differences in how multiple mentions of the same entity are coreferenced in reference versus hypothesis tagging inherently lead to cases where subsets of mentions that should refer to the same entity are split or cases where subsets of mentions that should correspond to different entities are merged. The scorer creates a bipartite graph between the sets of entities identified in reference vs. hypothesis documents, weighting each edge by the value score of the potentially corresponding pair. An algorithm then finds a set of matches that optimizes the aggregated value of these scores.

The scorer identifies potential correspondences between relation mentions in reference and hypothesis documents in a similar fashion. It prefers connections between relations that are *mappable*, where mappable is defined as having arguments that are matched entities. However it only requires one argument's mentions to have a 30% overlap in order to consider the relation to be initially *matched*. Note, however that it does not typically prefer such matches to those whose relations are fully mappable (which likely have a higher value score), and it does not count partially mappable relations as correct in its final results. As with entities, the scorer creates a bipartite graph between the sets of relations identified in the reference vs. hypothesis documents, and an algorithm finds a set of matches that optimizes the sum of the value scores assigned to each edge.

The procedure described above applies a rather loose and imprecise set of criteria for determining whether an extractor has correctly identified an entity or relation. An extractor could meet these requirements and still produce results that would be unacceptable for use without additional correction. However, the most conservative alternative, requiring that mentions and tags be completely identical to count as a match, may be overly rigid in some circumstances, especially considering the manifold attributes entities and relations have and the degree of imprecision that is inherent in natural language. Thus, we have found it is useful to analyze results using multiple sets of criteria that impose different degrees of rigor in their requirements for accepting a match identified by the ACE scorer [Goldstein-Stewart and Winder 2009]. The first criterion requires type and subtype of matching reference and hypothesis entities or relations to be the same. The second increases the level of specificity by requiring all of the attributes of the reference and hypothesis to be the same, as well. Correctly determining entity "level" is an additional criterion for entity extraction (an entity has a "NAM" level if one of its mentions includes a named mention, and a "NOM" level if there is no named mention but there is a nominal mention). Finally, having no reference arguments missing from the hypothesis is an additional criterion for relation extraction.

Table 1 and Table 2 display a spectrum of precision, recall, and F1 scores for our NW documents; these were generated by applying different combinations of the criteria described above to comparisons between tags generated by ST and RB vs. those created by humans. In this context, precision refers to the ratio of correctly extracted assertions to total number of extracted assertions, recall refers to the ratio of correctly extracted assertions to the total number of assertions that should have been extracted, and F1is the harmonic mean of precision and recall [2 * (precision * recall)/(precision + recall)]. In keeping with our earlier results [Goldstein-Stewart and Winder 2009], precision and recall varied considerably as the requirements for establishing correspondence changed. Yet it is interesting that even under the most naïvely optimistic measure of how well RB and ST performed (i.e. requiring only a match), extraction was still quite errorful, particularly for relations.

Table 1. Precision (P) , recall (R) , and F1 scores with different criteria for matching reference and hypothesis entities (scorer's unweighted results shown in bold)

Criteria for Being Found	ST P	ST R	ST F1	RB P	RB R	RB F1
matched	78.6	84.5	81.4	77.8	87.6	82.4
matched, same type/subtype	63.3	67.9	65.5	66.3	74.7	70.3
matched, same attributes	51.4	55.2	53.2	53.7	60.5	56.9
matched, same level	75.1	80.7	77.8	74.3	83.7	78.7
matched, same type/subtype, same level	61.4	66.0	63.6	64.2	72.3	68.0
matched, same attributes, same level	**49.8**	**53.5**	**51.6**	**51.9**	**58.5**	**55.0**

Table 2. Precision (P) , recall (R) , and F1 scores with different criteria for matching reference and hypothesis relations (scorer's unweighted results shown in bold)

Criteria for Being Found	ST P	ST R	ST F1	RB P	RB R	RB F1
matched	79.4	50.4	61.7	72.4	53.2	61.4
matched, same type/subtype	65.2	41.4	50.7	58.4	42.9	49.5
matched, same attributes	53.9	34.3	41.9	47.1	34.6	39.9
matched, arg1&2 not missing (i.e. *mappable*)	50.7	32.2	39.4	51.1	37.6	43.3
matched, no missing ref. args	49.0	31.1	38.1	48.9	35.9	41.4
matched, same type/subtype, arg1&2 not missing	42.0	26.7	32.7	41.9	30.8	35.5
matched, same type/subtype, no missing ref. args	40.9	26.0	31.8	39.8	29.3	33.8
matched, same attributes, arg1&2 not missing	35.4	22.5	27.5	33.8	24.9	28.7
matched, same attributes, no missing ref. args	**34.8**	**22.1**	**27.0**	**32.8**	**24.1**	**27.8**

We found similar ranges of measures when we examined extraction performance for the other genres in the corpus. While their exact values differed, they uniformly shared the feature that under even the least stringent

requirements for matching, precision and recall were still significantly below 100% (Table 3 and Table 4), and that when additional criteria were considered, performance worsens considerably. Thus, it is unlikely that operational users would allow extraction results generated by either ST or RB to populate a knowledge base without a human's correcting the extraction errors first. Below, we consider the actions in SW that might be required to address various types of extraction errors observed with this corpus.

Table 3. Precision (P) , recall (R) , and F1 scores for matching reference and hypothesis entities across different genres (assumes most lenient matching criteria)

Corpus	ST P	ST R	ST F1	RB P	RB R	RB F1
BC	68.9	78.2	73.3	73.9	79.1	76.4
BN	84.0	83.7	83.8	79.8	89.1	84.2
CTS	87.6	67.4	76.2	79.1	69.1	73.8
NW	78.6	84.5	81.4	77.8	87.6	82.4
UN	73.6	74.2	73.9	73.5	73.9	73.7
WL	75.3	81.5	78.3	75.5	84.4	79.7

Table 4. Precision (P) , recall (R) , and F1 scores for matching reference and hypothesis relations across different genres (assumes most lenient matching criteria)

Corpus	ST P	ST R	ST F1	RB P	RB R	RB F1
BC	76.7	41.6	53.9	66.9	46.8	55.1
BN	84.1	45.8	59.3	77.3	55.8	64.8
CTS	81.2	43.7	56.8	67.9	45.6	54.5
NW	79.4	50.5	61.7	72.4	53.2	61.3
UN	82.3	37.2	51.3	76.7	40.9	53.3
WL	72.2	44.0	54.7	67.2	49.1	56.7

3.2 Accuracy and Errors: Significance for SW

As discussed above, neither ST nor RB extracted entities and relations from our corpus with sufficient accuracy to permit automatic knowledge base population. Thus, both ST and RB would require a human user to review and correct results before committing them to the database. Naturally, the specific actions a user would have to take to process extraction results would depend on the system that is developed to support this kind of workflow. We modeled SW after our FEEDE prototype, which converts ACE relations into somewhat simplified form that matches the data model of the knowledge base it populates. FEEDE treats both entities and relations as consisting of a single attribute – type – which is merged from the type/subtype attribute pairs that appear in ACE. Relations also consist of a subject entity and an object entity (or argument-1 and argument-2), and entities are further defined by the extent of the text their mentions encompass. Although other systems might approach the extraction review/knowledge base population problem in slightly different ways, we believe FEEDE captures the basic functionality that any semi-automated extraction/population system would need to provide; also, as semi-automatic knowledge base population is a relatively new concept, there are few other reference systems on which we could base SW. Thus, we believe our characterization of the SW actions required to correct extraction errors in FEEDE provides a useful benchmark.

First, we consider the range of extraction errors that require user responses in SW. Most errors are specific cases or combinations of cases of precision errors. The extractors may generate spurious relations that are not correctable to any actual relation in the text. There may be errors in the relation type and in the type of the entities that make up the relation arguments, as well as errors in the extents of their representative mentions (cases of too much or too little text extracted). Finally there may be cases where one or both of the arguments are incorrect or missing, which can be divided into cases where the required entity argument (a) has been found in another relation but simply not applied to this relation, or (b) has been completely missed. Note that while these last two do not differentially impact the error rates as computed by the ACE scorer, they require significantly different correction activities.

Note that our analysis intentionally excludes those cases in which the extractors miss a relation completely. In reality, FEEDE does support so-called relation recall correction: If a user finds a non-extracted relation in the evidence text, s/he has the option to add it manually to the knowledge base. Unfortunately, such recall correction is extremely ad hoc. If a mention corresponding to a non-extracted relation happens to appear in close proximity to a mention corresponding to an extracted relation the user is reviewing, s/he may detect and ultimately record it; however, there is no way of knowing a priori whether a user will actually read the necessary evidence in its full context and realize that the relation is missing. Clearly, the likelihood of this occurring decreases sharply as the proximity of the missed relation from a found relation decreases. Ultimately, the only way to ensure users detect all such relation recall errors would be to require them to read through each document in its entirety, which would

defeat the purpose of using a semi-automated extraction system to facilitate knowledge discovery. We assume that users of a semi-automatic extraction system will know the recall error rates for the relation types of interest to them and will accept these as a precondition for using the system. Therefore, for the purposes of our analysis, we will only consider how precision errors impact time required to review and correct those relations that ST or RB actually extract.

For any correctable assertion, there are five potential elements a user may need to edit: the first argument's text, the first argument's type, the second argument's text, the second argument's type, and the relation type. Users correct entity and relation type in SW by selecting an appropriate alternative from a pull-down list. How users correct the argument text depends on the nature of the error, however. An extractor may correctly identify an entity in a relation but incorrectly extract the entity's PEN, which FEEDE displays in place of the specific entity mention that is an argument of the relation mention, since it is the PEN that FEEDE ultimately uploads to the database. PEN errors generally involve the extractors' incorrectly determining the boundaries of the mention from which the PEN is extracted (i.e., the extracted extent includes too much or too little text). In this case, users must manually edit the argument by copying and pasting the correct text from the evidence document (FEEDE also lets them type corrections directly into the argument field, but to simplify the analysis we assume users always copy/paste to ensure accuracy); although users perform this corrective action for a specific argument in a specific assertion, FEEDE lets them apply the correction to all entities coreferenced to the PEN. In other cases, an extractor may incorrectly identify one or both entities in the relation. If it extracted the correct entity elsewhere in the document, the PEN will appear within a pull-down list containing alternative entities of the same type, and the user can correct the assertion by selecting the appropriate PEN from the list. If the extractor did not discover the correct entity anywhere within the document, the user must manually copy and paste the correct text from the document evidence into the argument field. However, when the user corrects an assertion by adding an un-extracted entity in this manner, FEEDE adds the new entity to the list of possible alternatives available for that document. Therefore if the extractor missed this entity in other assertions, the user can correct these other assertions by selecting the added entity from the pull-down list.

Tables 5-7 display the rate at which ST and RB produce these different classes of errors across the six genres in our corpus: Table 5 shows the rate of entity type errors, Table 6 shows the rate of relation type errors, and Table 7 shows the rate of extent errors. Note that many of these error rates overlap with one another as errors can and do co-occur.

Table 5. Rate of entity type errors out of all relations (correctable and spurious) returned in automatic tagging for argument 1 or argument 2 of the relation.

System/Argument	BC	BN	CTS	NW	UN	WL
ST/Arg1	0.049	0.060	0.000	0.038	0.000	0.010
ST/Arg2	0.039	0.053	0.035	0.055	0.048	0.041
RB/Arg1	0.030	0.050	0.019	0.038	0.041	0.043
RB/Arg2	0.098	0.088	0.038	0.075	0.055	0.095

Table 6. Error rates for relation type/subtype over relations returned by the system (a. excludes spurious relations while b. includes spurious relations, which do not contribute to the type/subtype errors)

System	BC	BN	CTS	NW	UN	WL
ST (a.)	0.169	0.162	0.139	0.167	0.094	0.111
ST (b.)	0.146	0.149	0.118	0.142	0.065	0.082
RB (a.)	0.234	0.194	0.145	0.183	0.185	0.200
RB (b.)	0.188	0.153	0.132	0.140	0.164	0.129

Table 7. Number of extent errors out of all relations (correctable and spurious) returned in automatic tagging for the PENs representing argument 1 or argument 2 of the relation.

System/Argument	BC	BN	CTS	NW	UN	WL
ST/Arg1	0.000	0.010	0.000	0.006	0.016	0.000
ST/Arg2	0.010	0.073	0.000	0.043	0.048	0.010
RB/Arg1	0.068	0.028	0.000	0.018	0.014	0.000
RB/Arg2	0.038	0.045	0.000	0.045	0.068	0.009

All of the extraction errors described create additional work for the user who must identify and correct them; however, different types of errors require different kinds of corrective actions which differentially contribute to the user's overall workload. Having determined the frequency with which different errors occurred in our extraction

results and identified the SW actions that would be required to correct them, we now present an analysis of how much more efficient it would be to find and enter information from our corpus using SW vs. MW, using our RB and ST extraction results as inputs to a model-based GOMSL workflow analysis.

4. WORKFLOW ANALYSIS

4.1 Modeling Approach

We built GOMSL models for SW and MW and used them to conduct simulation-based analyses in GLEAN, using the methodology described in [Haimson and Grossman, 2009]. GOMSL is an executable version of the Natural Language GOMS (NGOMSL) formalism developed by [Kieras 1988]. GOMSL models define sequences of perceptual, cognitive, and motor activities that a user would perform to accomplish tasks in a GUI. GLEAN reads GOMSL models and simulates a user carrying out these activities. GLEAN is a simplified cognitive architecture. It represents the core components of human computational machinery: cognitive processing, attention and perception, perceptual buffers and working memory, long term storage of information, and motor responses. GLEAN also includes a "device simulator", which represents the modeled GUI with which the simulated user interacts. GLEAN computes task completion time using the standard set of empirically validated GOMS operators that specify various cognitive, perceptual, and motor activities (e.g., adding or retrieving information from working memory, scanning a visual interface for information, moving a mouse, or typing text; see [Card et al. 1983; Kieras 2006]).

GOMSL models define a set of methods that act upon declarative information objects with slots and values. Objects may be items in working memory or long-term storage, or they may be percepts, including visual objects that describe characteristics of the modeled GUI. Figure 3a shows two objects from the MW model. They describe the first task to be performed (open file containing documents) and the visual representation of the file in the file folder.

a.

```
Task_item: task1
  NAME is First.
  TYPE is Open_file.
  FILENAME is file1.
  NEXT is task2.

Visual_object: file1
  TYPE is file.
  BELOW is 2.
```

b.

```
MFG: Open File
  Step 1. AG: Choose Item_from_set using FILE_NAME of
    <current_task>, and file1.
  Step 2. AG: Double_select Object using <found_item>.
  Step 3. Delete <found_item>; RGA.

MFG: Double_select Object using <obj>
  Step 1. Look_at <obj>.
  Step 2. Point_to <obj>.
  Step 3. Verify position_is_correct.
  Step 4. Double_click mouse_button.
  Step 5. Delete <obj>; RGA.
```

Fig 3. Objects (a) and methods (b) from MW model.

GOMSL methods describe the sequences of perceptual, cognitive, or motor operators that accomplish the various activities into which tasks may be hierarchically decomposed. Methods define activities at different levels of granularity; higher level methods accomplish more general goals and typically include steps that execute lower-level methods accomplishing more specific goals. When more than one method is available to accomplish a given goal, GOMSL models include "if… then…" selection rules that specify the conditions under which one method should be chosen over another. Methods can also include "if… then…" decision rules. GLEAN executes the steps of each method in the order in which they are defined.

Figure 3b shows two methods from the MW model. The first, `Open File`, accomplishes the goal of opening the file of documents (`MFG` = Method for Goal), which involves spawning two subgoals (`AG`= Accomplish Goal) to (a) visually search the contents of a file folder until the icon corresponding to the desired file is located (`Choose Item_from_set`), and (b) move the mouse to that icon and double click to open it (`Double_select Object`). Note that `Choose Item_from_set` and `Double_select Object` are reusable methods that `Open File` calls using specific arguments, which may be obtained from working memory. For example, `Choose Item_from_set` uses the name of the file that GLEAN obtained by accessing the task `task1` and storing its contents in working memory, including the value stored in the `FILENAME` slot). The second method accomplishes the `Double_select Object` goal, which `Open File` spawns. Note that `Double_select Object` takes an `<obj>` argument, which `Open File` fills with `<found_item>`, a working memory pointer to the file icon that `Choose_item_from Set` finds. `Look_at`, `Point_to`, `Verify`, and `Double_click` are primitive

GOMSL operators that each have an empirically-determined completion time. Each step consumes an additional .050 sec of cognitive processing time, including the final step that deletes unneeded items from working memory and finishes the method (`RGA` = Return with Goal Accomplished).

We built models for MW and SW, including minimal representations of the GUI elements required for executing both workflows. The models are similar to those that [Haimson and Grossman 2009] developed; however, they based their SW model on an earlier version of FEEDE and compared their semi-automated workflow with a somewhat more complex manual workflow based on an actual operational system. We chose to simplify our MW model to eliminate unnecessary procedures that might unnecessarily increase task completion time. Also, as in [Haimson and Grossman 2009], we chose to model error-free human performance, partly because we lacked sufficient empirical or theoretic grounds to estimate types and frequencies of human errors but also because we wanted to assess workflow efficiency given the upper limits of human performance.

Our models follow the standard GOMS unit task structure [Card et al. 1983; Kieras 2006]. Higher level goals are decomposed into a few basic tasks represented by task memory instances that form a linked list. The models loop through methods that accomplish these tasks, which are themselves decomposed into subtasks, some of which are shared across methods and models (e.g., basic activities such as finding a button to click or scanning a pulldown menu). The models capture the basic cognitive, perceptual, and motor activities that comprise SW and MW although we simplified our representation of the more complex cognitive activities when it seemed legitimate and expedient to do so.

One of the simplifying assumptions we made was that users can rapidly translate source text into the same format in which they represent entities and relations in working memory, enabling them to quickly determine whether or not relations match the entities and relations. In reality, reading comprehension is far more complex, and the process of determining whether or not assertions accurately capture source is far more nuanced. However, skilled readers inspecting text in a familiar format and on a familiar subject would be likely to comprehend the meaning of source text fairly quickly, such that the additional time required to translate assertions and source text into a common symbolic representation would contribute little to the overall task completion time. Thus, we believe the simplification did not affect our overall results and greatly facilitated model implementation.

Another simplifying assumption involved our representation of visual search through lists of items. As in [Haimson and Grossman 2009], we modeled visual search through lists using the GOMSL `Look_at` operator and wrote methods that iteratively focused on an object, compared its properties with those of the sought-after target, and then either terminated search (if there was a match) or continued by focusing on the next object to be evaluated. This captures the basic seriality often observed in menu search [e.g., Byrne et al. 1999]. To simulate a user skipping over groups of items with the same beginning letter in an alphabetically-organized list, we assumed that each group of items that begins with the same letter constitutes a single item to be searched. Clearly, the number of groups that a user needs to search through varies from situation to situation; to simplify analysis, we assumed that users find the item they are looking for after rejecting an initial group of "distractor" items.

A final example of a modeling simplification involves our representation of the SW and MW GUIs. We modeled GUI screens as static collections of individual components using the default formalisms available in GOMSL. GLEAN does include an API that enables a model simulation to be run in conjunction with a separate C++ device simulation. However, we opted for the simpler approach of creating representations that encompassed all of the states an object could be in and writing GOMSL methods that only responded to aspects of the object that were appropriate for the state it was currently in (e.g., we defined a pulldown menu as an object containing both the list handle, which appears initially, and the list of items that appears below it when the user clicks the handle – the method that scanned the list was conditional on the method that selected the list handle). In this manner, we were able to recreate much of essential dynamic workflow without developing separate dynamic GUI simulations.

To analyze the full range of activities in each workflow, we created different GUI models that contained visual representations of each distinct type of data (e.g., a correct assertion, an assertion with an incorrect subject entity that could be corrected via a pulldown, an assertion with an incorrect subject entity that could only be corrected via copy and paste, etc.). We simulated SW and MW using each of these GUI models to compute task completion time estimates for each of the cases users might encounter. Note that GOMSL does not currently allow modelers to specify the physical location and size of visual objects without coding up a separate C++ simulation; thus, in computing task completion times, GLEAN used a default mouse movement time of 1.1 sec instead of calculating movement time by Fitts' Law [Card et al. 1983].

We used the results of these simulations to derive equations characterizing task completion time under the full range of conditions users may experience executing MW or SW, adding in estimates of reading time by assuming that users read two words per gaze [Card et al. 1983] and using GLEAN's estimate of .3 sec to shift, focus, and process each gaze. We then used these equations to derive measures of the efficiency with which users could

discover and enter data from our corpus using MW or SW. Note that the equations calculate the time users would spend doing actual work; they do not account for lags in system response, which would contribute to the total time required to complete the task. However, in practice, uploading and downloading information from a knowledge base account for the largest waiting times, and both SW and MW would be susceptible to these.

4.2 Workflow Completion Time Equations

Based on the results of our GLEAN simulations and our analysis of extraction performance and how it impacts FEEDE, we derived two equations to estimate time to complete either SW or MW, given the characteristics of the corpus to which they are applied. The equations are as follows (all constants are in seconds):

Workflow_Completion_Time_for_MW

Workflow_Completion_Time_for_SW=

:

- $\boldsymbol{E1_a}$ = number of unique entities that appear as the subject argument in a relation per document for the automatically-annotated corpus
- $\boldsymbol{E1_m}$ = number of unique entities that appear as the subject argument in a relation per document for the human-annotated corpus
- $\boldsymbol{E2_m}$ = number of unique entities that appear as the object argument in a relation per document for the human-annotated corpus
- $\boldsymbol{E1_{mx}}$ = number of repeated mentions of entities that appear as the subject argument in a relation per document for the human-annotated corpus
- $\boldsymbol{E2_{mx}}$ = number of repeated mentions of entities that appear as the object argument in a relation per document for the human-annotated corpus
- $\boldsymbol{R_a}$ = number of unique relations per document for the automatically-annotated corpus, including both relations that are correctable and those that are completely spurious
- $\boldsymbol{R_a'}$ = number of unique relations per document for the automatically-annotated corpus that are matched and correct or correctable to human-annotated relations (i.e., relations that can ultimately be uploaded to the database)
- $\boldsymbol{R_a}$ = number of relations per document for the automatically-annotated corpus that for all mentions of the relation have only pronouns in either argument
- $\boldsymbol{R_m}$ = number of unique relations per document for the human-annotated corpus
- $\boldsymbol{R_{mx}}$ = number of repeat mentions per document of relations for the human-annotated corpus
- $\boldsymbol{D}$ = number of documents inspected
- $\boldsymbol{B}$ = number of batches
- $\boldsymbol{V_a}$ = varying per relation work time for SW
- $\boldsymbol{WS}$ = words per sentence out of all sentences
- $\boldsymbol{S}$ = number of sentences across all documents
- $\boldsymbol{WS_a}$ = words per sentence containing a relation for SW
- $\boldsymbol{WP_a}$ = words per paragraph containing a relation for SW

The equations combine costs associated with the various activities that are dependent upon the number and properties of batches, documents, sentences, relations, and entities, which we refer to collectively as corpus description variables. For MW, the costs that vary with the number of batches (12.35 sec per batch) include the overall time spent internally coordinating MW activities (e.g., keeping track of which task is being performed and which task needs to be performed next) and the specific time spent finding and opening the file containing the document batch. The MW costs that vary with the number of documents include 57.45 sec to find source information (document identification number, date, title, and classification) in the document header and copy/paste it into the appropriate field in the GUI, plus time to read all of the sentences in the document ($0.3(\text{RoundUp}(.5WS)S)$ and time to process relations identified in the text. The time to process relations includes the time the user requires to determine whether s/he has already recorded the relation from that document ($2.6(R_m+R_{mx})$); $14.2R_m$ sec to re-read

the sentence (to verify interpretation and relevance), highlighting the sentence containing the relation (to indicate the relation has been recorded), and commit it to memory (to allow the user to quickly determine whether s/he has entered the information if s/he encounters it later in the document); and additional time spent actually recording the relation. Finally, recording the assertion includes internally coordinating data entry (0.35 R_m), using pulldown lists to record relation type and argument type for subject and object entities (21.8R_m), and entering the names of the entities, themselves. We assume that users copy and paste the name of an entity from the source text to the appropriate field in the MW GUI the first time they see the name in the batch, which takes 12 sec. However, if they record additional relations in which the same entity appears, we assume users can type the first four letters of that entity's name into the appropriate MW GUI field, at which point the MW GUI auto-completes the entity name (this process is estimated to take 5 sec).

For SW, the costs that vary with number of batches include the overall time spent internally coordinating SW activities (e.g., keeping track of which task is being performed and which task needs to be performed next), which is estimated at 12.25 sec, plus 26.35 sec to upload a batch of documents (which includes naming the batch, classifying the batch, and browsing to the file to be uploaded) and additional time for evaluating the assertions that are extracted (including 6.85 sec of time to coordinate this subtask internally). Users must select each subject entity to view its assertions (3.55 sec) and then inspect the subject entity PEN, relation, and object entity PEN, all of which is estimated to require 7.95 sec per assertion. We assume users inspect the highlighted mentions in the source text and then read the full sentence in which the mentions appear, which requires 1.3 seconds to locate the sentence's beginning plus time to read all the words in the sentence (0.3(RoundUp(.5WS)). If the sentence does not provide sufficient evidence for the user to determine whether the assertion arguments are properly coreferenced (i.e., if the arguments mentions are pronouns or nominals from which the PEN cannot be inferred), then s/he must read the preceding text in the paragraph to find co-referencing information. For simplicity, we assume that such information exists within the same paragraph and that users need to read on average a quarter of the paragraph to find it, so we compute this cost as 1.3 seconds to locate the paragraph's beginning plus (0.3(RoundUp(.5WS)) applied to the words in the first quarter of the paragraph.

After inspecting an assertion and verifying co-reference, the SW user must decide whether or not the relation is correct. Based on the results of this decision, the user either accepts the assertion as is, corrects it, or ignores it completely. We account for the time to complete these actions with the V_a term, which varies depending on whether and how the user corrects the assertion. Note that V_a represents the total time users spend on these activities; if an assertion requires multiple actions to be performed, V_a for that assertion will be the sum of the times required to perform those actions. Table 8 indicates time required to perform different actions for an assertion, though there is an additional overhead cost of 0.35 seconds for each non-spurious case to account for the user deciding what correction is necessary, if any. An assertion can be completely correct, in which case the user simply verifies that s/he does not need to make any edits. It can also be completely spurious, in which case the user selects an "ignore" button and moves on to the next assertion. Otherwise, the user corrects relation type, one or both entity arguments, and/or one or both entity types as necessary. Users correct entity and relation types simply by selecting the correcting type in a pulldown lists; argument correction is somewhat more complex, however. If the user determines that an entity is correct but that its extracted PEN is too short or too long, s/he corrects the PEN by copying and pasting the correct name from the source document into the argument field (in reality, users might choose to type or delete characters within the argument field, as well, but we chose not to represent this alternative to streamline the model). If s/he determines that an entity is incorrect, s/he first examines a pulldown list of all PENs extracted from the currently-viewed document to see if the correct PEN appears in the list; if it does, s/he selects it, and if it does not, s/he copies and pastes the correct PEN from the source text into the argument field (again, we chose not to model an alternate method of typing the name directly into the argument field). If users correct a PEN by copying and pasting from the source text, they have the option to apply the correction to all entities co-referenced to the same PEN; users should choose this option if the entity is correct and they are merely correcting a PEN that is too long or too short, but they should not choose it if they are copying and pasting in a PEN for a new unextracted entity because the PEN may be correct for the other entities that are co-referenced to it.

Once the SW user determines that an assertion is not spurious and corrects it as necessary, s/he indicates that the assertion is valid by selecting a "validate" button (5.8 sec) and moves on to the next assertion linked to that subject entity. Once s/he is finished with all of the assertions linked to the current subject entity, she selects the next subject entity in the list and processes all of its assertions. Once s/he is finished processing all assertions extracted from the batch, s/he reviews the complete list of validated assertions and uploads them to the database, which imposes a base time cost of 7.65 sec plus an additional 1.65 sec per assertion.

Table 8. Different time costs for actions represented by V_a (in seconds). Note that V_a represents total time users spend on these activities; if an assertion requires multiple actions to be performed, V_a will be the sum of the times required to perform those actions.

Situation	Required Correction Actions	Time to perform action (in seconds)
Completely Correct Relation	None	0.30
Completely Spurious Relation	None (select "ignore" button)	3.00
Incorrect Entity Argument (Correct Entity Missing)	Copy-Paste	30.00
Primary Entity Name Extent Too Long/Too Short	Copy-Paste	17.95
Incorrect Entity Argument (Correct Entity Extracted)	Pull-down	13.15
Incorrect Entity or Relation Type	Pull-down	6.95

The workflow completion time equations estimate the time required to perform SW or MW given the characteristics of the corpus to which the workflows are applied (as reflected in the values of the corpus description variables). We now describe how we used these equations to estimate workflow completion time for our test corpus.

4.3 Workflow Completion Time Analysis for Test Corpus

We used the results of our extraction analysis to derive genre-specific values for each of the corpus description variables, assuming a single batch of documents (B=1). For each genre, we set D equal to the number of documents from that genre and calculated mean counts for the other corpus description variables based on document characteristics and extraction results. Table 9 shows these values for each of the six genres.

Table 9. Corpus description variables for the six different genres normalized for one document.

Variable	Tagger	Name	BC	BN	CTS	NW	UN	WL
Unique first arguments	Human	$E1_m$	13.89	5.23	14.17	11.09	6.77	5.95
Repeated first arguments	Human	$E1_{mx}$	7.22	2.26	12.17	4.88	3.77	2.42
Unique second arguments	Human	$E2_m$	14.44	5.41	21.83	9.97	8.08	6.21
Repeated second arguments	Human	$E2_{mx}$	6.67	2.08	4.50	6.00	2.46	2.16
Unique first arguments	ST	$E1_a$	7.56	2.96	5.83	6.53	3.00	3.26
Unique first arguments	RB	$E1_a$	8.00	3.53	6.33	7.06	3.54	3.37
All relations	Human	R_m	21.11	7.49	26.33	15.97	10.54	8.37
Repeated mentions of relations	Human	R_{mx}	3.56	1.15	5.50	1.56	2.38	1.00
All relations	ST	R_a	11.44	4.08	14.17	10.15	4.77	5.11
All relations	RB	R_a	14.78	5.41	17.67	11.74	5.62	6.11
Correctable relations	ST	R_a'	8.78	3.43	11.50	8.06	3.92	3.68
Correctable relations	RB	R_a'	9.89	4.18	12.00	8.50	4.31	4.11
Pronoun arg. relations	ST	R_a''	2.11	0.53	10.33	0.74	1.85	1.37
Pronoun arg. relations	RB	R_a''	5.33	0.82	13.33	0.88	2.31	1.95
Sentences	N/A	S	60.44	9.58	140.00	15.91	77.85	27.63
Words/sentence	N/A	WS	13.43	13.77	8.68	18.69	11.49	13.43
Words/sentence holding relations	ST	WS_a	24.65	21.92	20.55	27.00	24.71	28.37
Words/sentence holding relations	RB	WS_a	22.47	22.46	18.72	28.88	24.42	29.32
Words/paragraph holding relations	ST	WP_a	128.28	112.62	34.56	31.03	64.07	71.09
Words/paragraph holding relations	RB	WP_a	131.88	113.79	34.55	33.89	61.95	76.46

To calculate average V_a for our corpus, we first determined V_a for every possible combination of activities SW users could perform to accept, correct, or ignore an assertion. Next, we determined how often each corresponding combination of extraction errors occurred for each of our genres. The Appendix shows these values for ST (Table A13) and RB (Table A14). By multiplying the time cost for a given combination of V_a activities by the frequency with which the corresponding combination of errors occurred, we derived initial weighted average V_a values for each of our genres.

These estimates did not account for the fact that some of the actions SW users take to correct entities can apply to more than one assertion, however. For example, if a user decides than an extracted PEN is too short or too long (an extent error), s/he can correct it and apply those corrections to every assertion in which that PEN appears. Similarly, if the user corrects an entity's type, s/he can apply that correction to every assertion in which the entity appears, as well. Finally, if a user determines that an assertion has an incorrect entity and that the pulldown list does not contain the correct choice (i.e., the extractor did not find the entity anywhere in the document), s/he can copy and paste the PEN from the source text into the assertion argument field; at this point, FEEDE adds the new PEN to the pulldown list, which means the user will be able to select the PEN from the list if s/he needs to add the missing

entity to any other assertions. To correct for these cases, we determined the frequency with which ST and RB extraction results contained extent errors, missing entities, or incorrect entity types for each genre (Table 10). We multiplied the frequency of extent errors by 17.95 (the time required to correct an extent error), the frequency of entity type errors by 6.95 (the time required to correct entity type), and the frequency of missing entities by 16.85 (equivalent to removing a copy-paste and adding a pull-down) and subtracted the results from our original average V_a scores for each genre. Table 11 shows the resulting final V_a scores.

Having derived genre-specific corpus description variables and V_a estimates, we could now compute workflow completion times for SW and MW to compare their relative efficiency for our test corpus. We described this analysis and our results next.

Table 10. Frequency of extent, entity type, and missing entity errors in ST and RB extraction results for each genre.

System	Redundant Error Type	BC	BN	CTS	NW	UN	WL
ST	Extent Error	0.000	0.020	0.000	0.014	0.000	0.000
ST	Entity Type Error	0.000	0.003	0.000	0.003	0.000	0.000
ST	Missing Entity	0.019	0.043	0.000	0.014	0.032	0.000
RB	Extent Error	0.045	0.013	0.000	0.010	0.014	0.000
RB	Entity Type Error	0.000	0.180	0.013	0.333	0.000	0.027
RB	Missing Entity	0.090	0.033	0.075	0.010	0.027	0.000

Table 11. Final V_a scores computed from combining GOMSL workflow results and observed ST and RB extraction errors.

Term	BC	BN	CTS	NW	UN	WL
V_a for ST	10.22	13.67	10.39	12.06	13.46	10.01
V_a for RB	15.23	11.30	13.79	10.31	17.34	11.88

4.4 Workflow Completion Time Analysis Results

Using the corpus description variables and final V_a scores described above, we calculated per-relation workflow completion times for each genre given MW, SW with RB as the extractor (which we will refer to as "SW_RB"), and SW with ST as the extractor (which we will refer to as "SW_ST"). We chose to examine workflow completion time on a per-relation basis instead of across the entire batch because extraction recall errors cause SW users to process fewer relations than MW users; using total batch completion time as a measure, SW might appear to be more efficient than MW simply because users process fewer assertions rather than because users process each assertion faster. Note that the per-relation workflow completion time measure distributes time costs that depend on variables other than number of relations (e.g., number of batches or documents) across relations included in the measure. For example, SW users incur a time cost for uploading batches to FEEDE, and MW users incur a time cost for recording document source information. The more relations per document, the less impact document-level costs have on per-relation workflow completion time, and the more relations per batch, the less impact batch-level costs have on per-relation workflow completion time. Thus, in addition to penalizing MW and SW for batches with fewer documents and shorter, sparser documents, the per-relation workflow completion time measure also penalizes SW for extractor recall errors.

To compute per-relation workflow completion time for MW, we divided total workflow completion time by the number of relations human annotators identified in the source text (i.e., R_mD). To compute per-relation workflow completion for SW, we divided the total workflow completion time by the number of correct or correctable relations extracted from the document (i.e., $R_a'D$). We also calculated two additional sets of per-relation workflow completion times. For MW, we computed per-relation workflow completion times excluding time spent reading each document (MW_no_reading); we were interested in determining how much of the MW completion time cost depended on time to enter the information alone (i.e., without additional time spent reading to discover the information to be entered). For SW, we computed per-relation workflow completion times assuming 100% extractor recall and precision (SW_100%_r&p); we were interested in determining how efficient SW would be if automation performed perfectly. We computed these values twice, once assuming a single batch containing all 152 of our documents, and once assuming that each document was its own batch.

Figures 4 and 5 display the results of these calculations. As shown, MW had the highest per-relation workflow completion time across genres, and SW_100%_r&p had the lowest, with MW_no_reading falling just under MW and SW_RB and SW_ST lying between MW_no_reading and SW_100%_r&p. As shown, genre did appear to affect per-relation workflow completion time, although it did not fundamentally alter the differences between workflows. For 152-document batches (Figure 4), the effect was largest for MW, smaller but still pronounced for

MW_no_reading and SW_RB, and largely absent for the SW_ST and SW_100%_r&p. In contrast, the effect appeared to be strong across all workflows for single document batches (Figure 5).

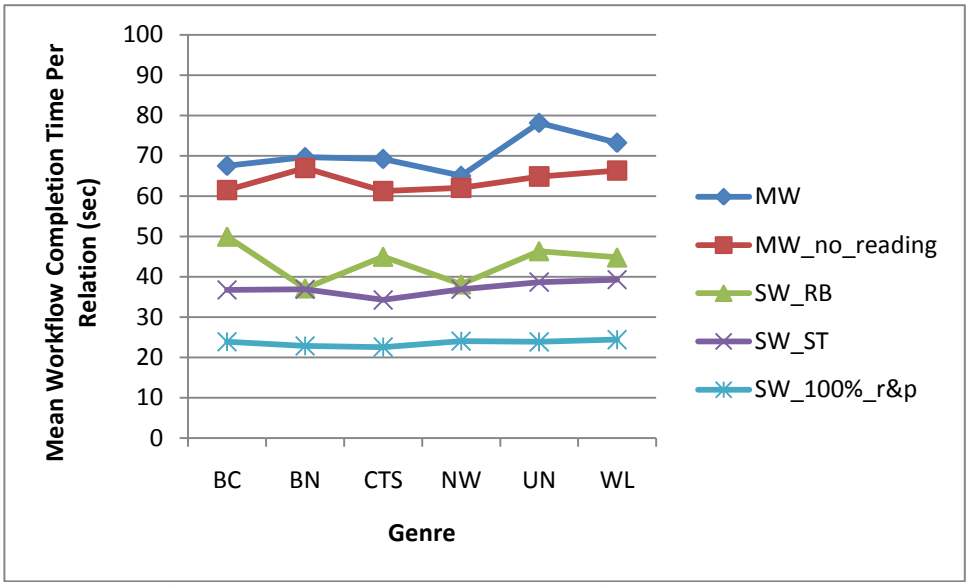


Fig 4. Average workflow completion time per relation (sec) computed for each genre (batch size =152 documents).

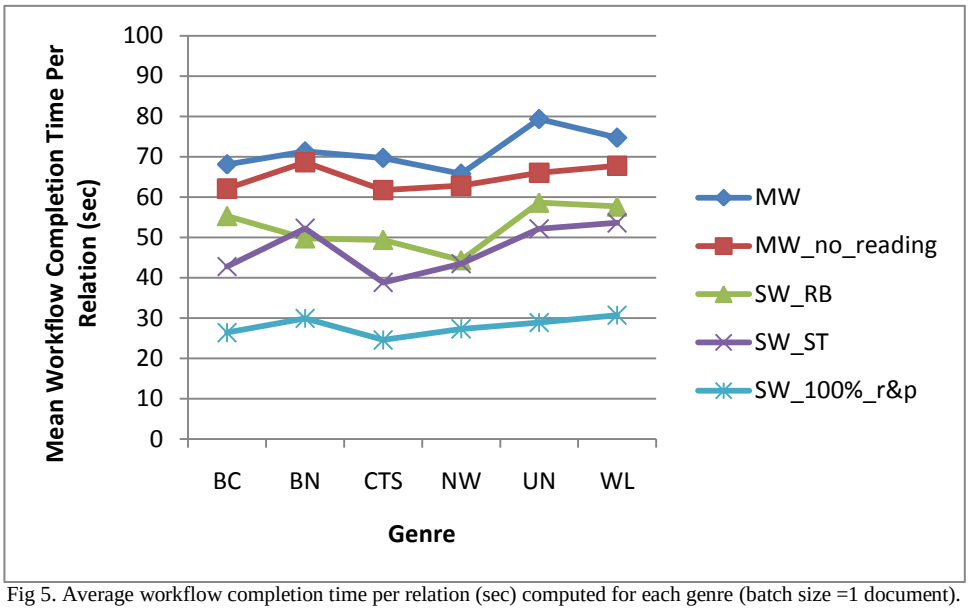


Fig 5. Average workflow completion time per relation (sec) computed for each genre (batch size =1 document).

The effect of genre primarily reflects differences in the number of relations processed per document and across an entire batch. As noted above, the per-relation workflow completion metric divides document-level time costs across all of the relations in a document, and batch-level time costs across all relations in a batch. The smaller the number of relations per document (which can result from both smaller documents and fewer relations per document), the greater the impact of document-level time costs. Note that BN, UN, and WL have smaller numbers of relations (both human annotated and extracted) than BC, CTS, and NW. As noted, systematic differences in the density of relations per document may partly account for overall differences in number of relations. Table 12 shows the average percentages of words in a document that are involved in a relation identified by human annotators, ST, or RB. Note that BN and NW are at least 50 percent denser than the other genres for ST and RB.

Table 12. Average ratios of text containing the extent of a relation to total text per document across six genres for three different relation annotation methods.

Tagger	BC	BN	CTS	NW	UN	WL
Ref	0.235	0.258	0.263	0.384	0.077	0.196
ST	0.053	0.085	0.016	0.127	0.012	0.041
RB	0.086	0.145	0.057	0.212	0.022	0.079

Document-level time costs have the largest impact on MW and MW_no_reading, since users have to record document source for both workflows. Users have to read through documents in MW, as well, which also contributes to document-level time (larger documents with fewer relations will have the greatest impact on per-relation reading time). In contrast, batch-level time costs have the largest impact on SW_RB, SW_ST, and SW_100%_r&p, since users have to upload batches of documents to FEEDE. Batch-level costs are minimal for large batches that produce a large number of relations (Figure 4), but they clearly impact workflow efficiency for small batches (Figure 5), such that SW_RB, SW_ST, and SW_100%_r&p all exhibit effects of genre that are roughly comparable to those of shown by MW and MW_no_reading, albeit for different reasons. Note that ST had a somewhat lower recall rate than RB, meaning that SW_ST results reflect a smaller number of relations than RB; this suggests that SW_ST's apparent advantage over SW_RB resulted entirely from ST's making fewer precision errors than RB.

To determine the statistical significance of these patterns, we recomputed per-relation workflow completion times given MW, MW_no_reading, SW_RB, SW_ST, and SW_100%_r&p for each document in the corpus. If either extractor failed to extract any relations from a document, we excluded that document from analysis; the modified corpus contained 138 documents (|BC|=9, |BN|=64, |CTS|=6, |NW|=34, |UN|=11, |WL|=14). We again computed these values twice, once assuming a single batch containing all 138 of our documents, and once assuming that each document was its own batch.

Figures 6 displays results of these calculations for a batch of 138 documents. MW had the highest per-relation workflow completion time across genres, and SW_100%_r&p had the lowest, with MW_no_reading falling just under MW and SW_RB and SW_ST lying between MW_no_reading and SW_100%_r&p. We performed a mixed design ANOVA with genre as a between-subjects variable and workflow (MW vs. MW_no_reading vs. SW_RB vs. SW_ST vs. SW_100%_r&p) as a within-subjects variable. There was a marginal effect of genre [$F(5,132)=2.1865$, $p=0.059$]. Mauchly's test indicated that results violated the assumption of sphericity for the repeated measures portion of the analysis [$W(4)=0.0634$, $p<.0001$]; thus, we applied a Greenhouse-Geisser correction to the degrees of freedom ($\varepsilon = 0.591$). Results indicated a significant main effect of workflow [$F(2.36,312.05)=30.02$, $p<.0001$] and a significant interaction of workflow and genre [$F(11.82, 312.05)=4.16$, $p<.0001$].

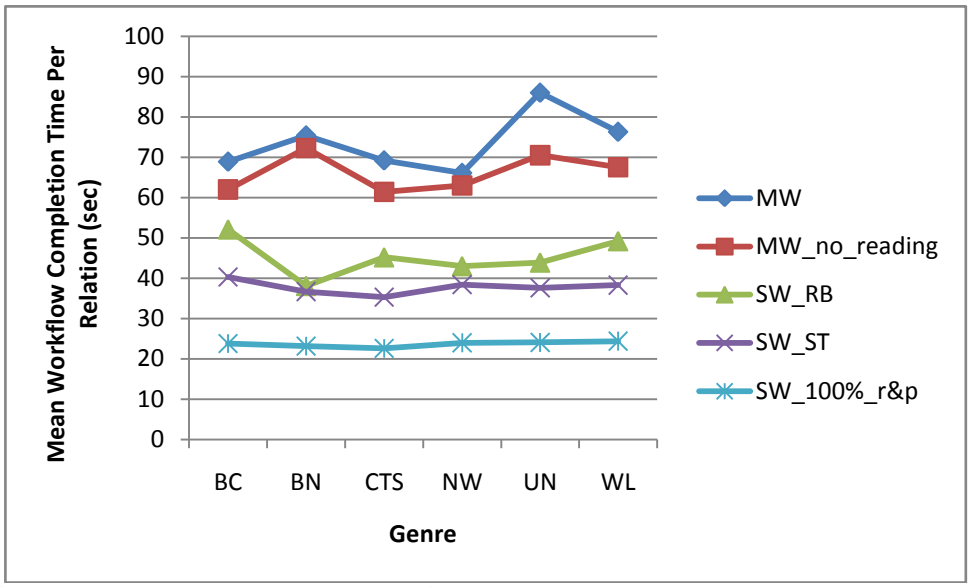


Fig 6. Workflow completion time per relation (sec) computed for each document and averaged across genres (batch size =138 documents).

As discussed above, the marginal effect of genre and interaction of genre with workflow largely reflects the impact of differences in amount of text and number of relations across genres. However, since differences in genre

did not appear to fundamentally alter differences between workflows, we examined the main effect of workflow further by averaging results across genre and conducting a series of Bonferroni-corrected t-tests and effect size tests, which yielded the following results:

- The difference between SW_100%_r&p and SW_ST was significant [$t(137)=-12.94$, $p<.0001$] with a large effect size [Cohen's $d=-1.527$].
- The difference between SW_ST and SW_RB was significant [$t(137)=3.25$, $p<.0001$] with a small-to-medium effect size [Cohen's $d=0.352$].
- The difference between SW_RB and M_no_reading was significant [$t(137)=-18.33$, $p<.0001$] with a large effect size [Cohen's $d=-2.193$].
- The difference between M_no_reading and M was significant [$t(137)=-11.47$, $p<.0001$] with a small-to-medium effect size [Cohen's $d=-0.442$].

Figure 7 display the results of these calculations for batches of single documents. As shown, MW again had the highest per-relation workflow completion time across genres, and SW_100%_r&p had the lowest, with MW_no_reading falling just under MW and SW_RB and SW_ST lying between MW_no_reading and SW_100%_r&p. Unlike in Figure 6, SW_RB and SW_ST are much closer to each other and MW_no_reading. We subjected results to a mixed design ANOVA with genre as a between-subjects variable and workflow (MW vs. MW_no_reading vs. SW_RB vs. SW_ST vs. SW_100%_r&p) as a within-subjects variable. The effect of genre was significant [$F(5,132)=4.148$, $p=0.016$]. Mauchly's test indicated that results violated the assumption of sphericity for the repeated measures portion of the analysis [$W(4)=0.0115$, $p<.0001$]; therefore we applied a Greenhouse-Geisser correction to the degrees of freedom ($\varepsilon = 0.533$). Results indicated a significant main effect of workflow [$F(2.13,281.42)=18.81$, $p<.0001$] but an insignificant interaction of workflow and genre [$F(10.66,281.42)=1.32$, $p=.214$].

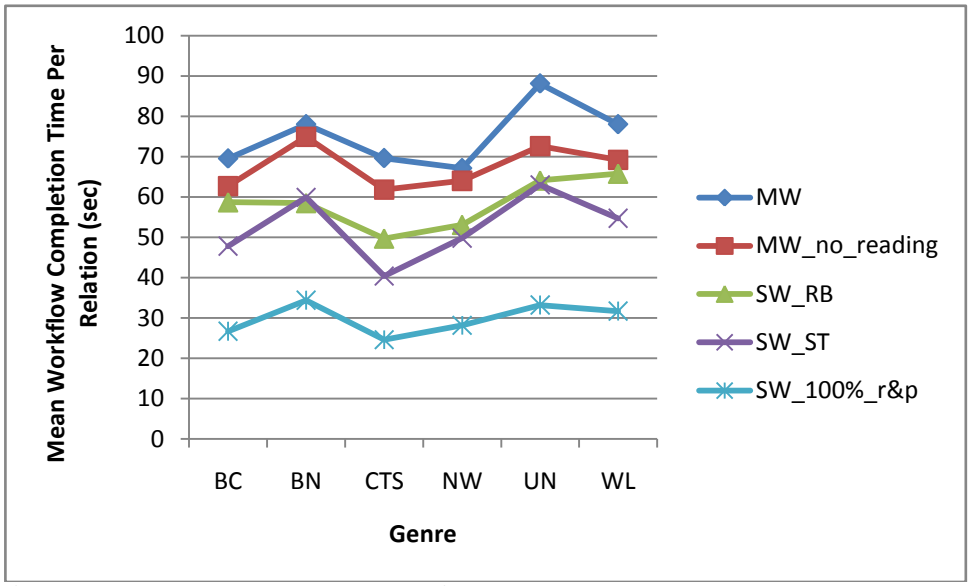


Fig 7. Workflow completion time per relation (sec) computed for each document and averaged across genres (batch size =1 document).

We examined the main effect of workflow further by averaging results across genre and conducting a series of Bonferroni-corrected t-tests and effect size tests, which yielded the following results:

- The difference between SW_100%_r&p and SW_ST was significant [$t(137)=-14.98$, $p<.0001$] with a large effect size [Cohen's $d=-1.490$].
- The difference between SW_ST and SW_RB was insignificant [$t(137)=-1.22$, $p<.2227$] with a small effect size [Cohen's $d=0.118$].
- The difference between SW_RB and M_no_reading was significant [$t(137)=-8.02$, $p<.0001$] with a medium-to-large effect size [Cohen's $d=-.710$].
- The difference between M_no_reading and M was significant [$t(137)=-11.47$, $p<.0001$] with a small-to-medium effect size [Cohen's $d=-0.380$].

The results of our workflow completion time analysis indicate significant effect of workflow. With a large batch, we estimated SW_ST and SW_RB to be roughly 1.5 to 2 times more efficient than MW, even when users do not have to read through documents for information discovery; if extractors perform without errors, we estimate the semi-automated workflow be 3 or even 4 times as fast as the manual one. However, with a batch of single documents, the automation advantage decreases because the efficiency of semi-automatic information discovery and entry is partly countered by batch-level time costs. We discuss these results and their implications for semi-automated information discovery and data entry next.

5. DISCUSSION

We performed a model-based analysis to quantify the efficiency of manual vs. semi-automatic information discovery/data entry workflows. We first evaluated the performance of rule-based (RB) and statistical (ST) extractors on our test corpus and demonstrated that the extractors made too many errors to allow for fully-automated population of a knowledge base, suggesting a need for semi-automatic systems like FEEDE. Recognizing that the efficiency of a semi-automatic workflow would depend on the efficiency of the actions a human user would take to review and correct extraction results, we mapped the different types of extraction errors to actions users would take to correct these errors in FEEDE and GOMSL to model the end-to-end workflow (SW). We developed a corresponding manual end-to-end workflow (MW) in GOMSL, as well, and then used GLEAN to derive estimates of time costs associated with the different modeled user activities. Next, we determined values for key corpus description variables for each of our document genres, calculated the frequency with which RB and ST made the different classes of correctable errors, and used these results to generate genre-specific per-relation workflow completion time estimates for different versions of MW and SW. Results quantify the estimated time saved using SW instead of MW (not accounting for potential differences in system response time) and demonstrate the dependence of these time savings on number of relations, batch size, and extraction accuracy.

We found that ST generated fewer precision errors than RB, which led to SW_ST outperforming SW_RB on the large batch. This advantage disappeared in the single document batch, however, because the high per-relation batch costs overshadowed differences in time required to correct relations. While these results do suggest that it would be preferable to use ST vs. RB for SW, it is important to note that RB has a higher recall rate than ST. If users are willing to sacrifice efficiency for completeness, RB may actually be the better choice. In fact, if it is critical to capture all or most of the information in document, it may actually be preferable to use MW instead of either SW_ST or SW_RB, even if it takes more time to complete the task. However, if users are willing to trade speed for completeness, SW with either ST or RB will increase their productivity considerably.

We believe that the synthesis of corpus analysis and workflow simulation provided a more complete picture of the advantage of inserting automation into the information discovery/data entry workflow. We could not have fully evaluated the utility of the automated capability without considering the complete system in which it is embedded. While the extractor analysis alone provided insight into the types of possible errors and the frequency with which they are produced, we could not appreciate the significance of those errors until we could quantify how they directly impact system use. This context is crucial considering that different types of errors impact the overall workflow in different ways; a minor extraction error (e.g., an extent error) might actually require more human effort to fix than a more significant extraction error (e.g., incorrect relation type).

We note that these methods did take some time and effort to execute. Our approach required expertise in the underlying system technology (content extraction in this case), GOMS, and advanced data analysis techniques. The corpus analysis required new scripts for computing and synthesizing results (although we leveraged the standard ACE scoring tools, they do not break results down to the level of detail we required). Devising the analysis scheme, coding the procedures, and computing results required several weeks. We were able to reuse the scripts with a few modifications for multiple sets of analyses, however, and we believe they could be easily repurposed to run additional analyses on new datasets. The GOMSL analysis required in-depth knowledge of the system workflow down to the level of individual keystrokes, both for understanding the specific keystroke and mouse sequences involved in each user task and, more importantly, the underlying cognitive processes. Detailed documentation and/or sufficient experience with the system would be necessary for gaining this level of familiarity (we were already well-acquainted with FEEDE before beginning this study). GOMSL modeling required several weeks, including writing the models and analyzing the simulation trace and metrics (GLEAN provides detailed summary statistics); GOMSL models could be modified and reused for similar applications. It was fairly straightforward to transform GOMSL results into workflow completion time equations, although it required a deep understanding of both the human workflow and the content extraction errors that gave rise to different workflow sequences. We developed additional scripts to run simulations using the content extraction results and workflow completion time equations. Finally, we used R to run final statistical analyses. All in all, we estimate that the entire process required approximately three

months to complete; we anticipate that future studies would take less time, depending on the extent to which we could reuse existing scripts and models.

Our analysis does assume that human-annotated documents represent ground truth and that human users will perform their roles in MW or SW with 100% accuracy. In reality, ACE annotations could include errors, which would directly impact extraction results. Moreover, MW users could miss some relations and/or enter them incorrectly into the knowledge base, and SW users could misinterpret assertions and/or edit them incorrectly, both of which would directly impact modeling results. Unfortunately, it is difficult to estimate the impact of human error on the extraction analysis or account for human error in our workflows models. However, ACE corpora do have relatively high rates of inter-annotator agreement. Moreover, there is no reason to believe that any one workflow would encourage a higher rate of human error than another; thus, while human errors could affect absolute measures of workflow completion time, they would be unlikely to change the overall pattern of results. Lacking a valid model of how and when people make errors on this task, we chose to model the upper levels of human performance; while this hurt the external validity of our results, it did allow us to obtain purer measures of the efficiencies and inefficiencies inherent in the workflows, themselves.

The workflow completion time estimates we generated are specific to the models we built to characterize MW and SW. We believe that our models are faithful representations of MW and SW, and we believe that SW and MW are reasonable representatives of semi-automated and manual workflows. It would be possible to develop alternate models of SW and MW that capture a slightly different set of human-computer interactions. It is likely that such models would produce somewhat different workflow completion time estimates. However, as long as the alternative models address the same fundamental information discovery and data entry tasks that our SW and MW models address, the overall patterns of results should be very similar to the ones reported here, assuming comparable corpus description variable values and levels of extraction accuracy. In this context, it should be noted that [Haimson and Grossman 2009] predicted somewhat larger advantages of using a semi-automated vs. manual workflow. However, they modeled their semi-automated workflow on an earlier version of FEEDE and modeled their manual workflow on a different manual data entry system than the one described here. More importantly, they defined less comprehensive workflow completion time equations than the ones described here, and they based their final analyses on idealized corpus description variable values and mock extraction results. It is likely that their estimates would more closely resemble those reported here had they followed the analysis process described above and used corpus description and extraction performance parameters derived from real-world data and extraction results.

We have demonstrated here that on a per relation basis, there is a definite advantage to SW over MW across several different text genres, an advantage that increases as the sizes of document batches, number of entities, and accuracy of extraction increases. While users will not find all of the relations in a source text using content extraction, they will be able to find a large number and enter them more rapidly into a knowledge base. Ideally, content extraction technology will mature to the point that user needs no longer review every result for accuracy. Until that time, assuming that users are willing to sacrifice recall, our results suggest that semi-automated information discovery and data entry may be a viable alternative to manual data entry for capturing structured information from raw text, making it possible to ensure data integrity while accelerating the human/system workflow.

ACKNOWLEDGEMENTS

We thank our sponsors, system developers, and subject matter experts for their support of this work. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, or the MITRE Corporation.

REFERENCES

ACE 2005. English annotation guidelines for entities version 5.6.1. Retrieved May 7, 2008 from: http://projects.ldc.upenn.edu/ace/docs/English-Entities-Guidelines_v5.6.1.pdf.

ACE 2005. English annotation guidelines for relations version 5.8.3. Retrieved May 7, 2008 from: http://projects.ldc.upenn.edu/ace/docs/English-Relations-Guidelines_v5.8.3.pdf.

BARCLAY, C., BOISEN, S., HYDE, C., AND WEISCHEDEL, R. 1996. The Hookah information extraction system. In *Proceedings of the Workshop on TIPSTER II*. ACL, Vienna, VA, 79-82.

BYRNE, M.D., ANDERSON, J.R., DOUGLASS S., AND MATESSA, M. 1999. Eye tracking the visual search of click-down menus. In *Proceedings of CHI'99*. ACM, New York, NY, 402-409.

CARD, S., MORAN, T., AND NEWELL, A. 1983. *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, Inc., Hillsdale, NJ.

DONALDSON, I., MARTIN, J., DE BRUIJN, B., WOLTING, C., LAY, V., TUEKAM, B., ZHANG, S., BASKIN, B., BADER, G., MICHALICKOVA, K., PAWSON, T., AND HOGUE, C. 2003. PreBIND and Textomy—mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC Bioinformatics*. 4:11.

GOLDSTEIN-STEWART, J., AND WINDER, R. 2009. Designing a system for semi-automatic population of knowledge bases from unstructured text. In *Proceedings of the 1st International Conference on Knowledge Engineering and Ontology Development (KEOD-IC3K)*. Funchal, Portugal, 88-99.

GRISHMAN, R., AND SUNDHEIM, B. 1996. Message understanding conference – 6: a brief history. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*. Copenhagen, Denmark, 466-471.

HAIMSON, C., AND GROSSMAN, J. 2009. A GOMSL analysis of semi-automated data entry. In *Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. ACM, Pittsburgh, PA, 61-66.

JOHN, B., AND KIERAS, D.E. 1996. The GOMS family of user interface analysis techniques: comparison and contrast. *ACM Transactions on Computer-Human Interaction*, *3*, 320-351.

KIERAS, D.E. 1988. Toward a practical GOMS model methodology for user interface design. In *The Handbook of Human-Computer Interaction* M. Helander Ed., Elsevier, Amsterdam, North-Holland, 135-158.

KIERAS, D.E. 2006. A guide to GOMS model usability evaluation using GOMSL and GLEAN4. Retrieved August 15, 2008 from (http://www.eecs.umich.edu/~kieras/goms.html).

KIERAS, D.E., WOOD, S.D., ABOTEL, K., AND HORNOF, A. 1995. GLEAN: A computer-based tool for rapid GOMS model usability evaluation of user interfaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, 91-100.

MARSH, E., AND PERZANOWSKI, D. 1998. MUC-7 Evaluation Of IE Technology: Overview of Results. Retrieved 2009 from: http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_toc.html.

MOENS, M.-F. 2006. *Information Extraction: Algorithms And Prospects For Retrieval*. Springer, Dordrecht, The Netherlands.

ACE 2005. The ACE evaluation plan. Retrieved Oct 14, 2009 from: http://www.itl.nist.gov/iad/mig/tests/ace/2005/doc/ace05-evalplan.v3.pdf.

VILAIN, M., SU, J., AND LUBAR, S. 2007. Entity extraction is a boring solved problem—or is it? In *HLT-NAACL – Short Papers*. Rochester, NY, 181-184.

8. APPENDIX

Table A1. Precision (P) , recall (R) , and F1 scores for BC corpus with different criteria for matching reference and hypothesis entities (scorer's unweighted results shown in bold).

Criteria for Being Found	ST P	ST R	ST F1	RB P	RB R	RB F1
matched	68.9	78.2	73.3	73.9	79.1	76.4
matched, same type/subtype	57.0	64.7	60.6	64.2	68.7	66.4
matched, same attributes	42.6	48.3	45.3	49.6	53.1	51.3
matched, same level	63.9	72.5	67.9	70.6	75.6	73.0
matched, same type/subtype, same level	53.0	60.2	56.4	61.7	66.1	63.8
matched, same attributes, same level	**39.9**	**45.3**	**42.4**	**47.6**	**50.9**	**49.2**

Table A2. Precision (P) , recall (R) , and F1 scores for BC corpus with different criteria for matching reference and hypothesis relations (scorer's unweighted results shown in bold).

Criteria for Being Found	ST P	ST R	ST F1	RB P	RB R	RB F1
matched	76.7	41.6	53.9	66.9	46.8	55.1
matched, same type/subtype	62.1	33.7	43.7	48.1	33.7	39.6
matched, same attributes	53.4	28.9	37.5	39.8	27.9	32.8
matched, arg1&2 not missing (i.e. *mappable*)	40.8	22.1	28.7	39.8	27.9	32.8
matched, no missing ref. args	40.8	22.1	28.7	36.8	25.8	30.3
matched, same type/subtype, arg1&2 not missing	34.0	18.4	23.9	31.6	22.1	26.0
matched, same type/subtype, no missing ref. args	34.0	18.4	23.9	28.6	20.0	23.5
matched, same attributes, arg1&2 not missing	30.1	16.3	21.2	25.6	17.9	21.1
matched, same attributes, no missing ref. args	**30.1**	**16.3**	**21.2**	**24.8**	**17.4**	**20.4**

Table A3. Precision (P) , recall (R) , and F1 scores for BN corpus with different criteria for matching reference and hypothesis entities (scorer's unweighted results shown in bold).

Criteria for Being Found	ST P	ST R	ST F1	RB P	RB R	RB F1
matched	84.0	83.7	83.8	79.8	89.2	84.0
matched, same type/subtype	68.6	68.4	68.5	68.8	76.9	68.6
matched, same attributes	53.9	53.7	53.8	52.9	59.1	53.9
matched, same level	80.0	79.7	79.8	76.6	85.6	80.0
matched, same type/subtype, same level	66.4	66.2	66.3	66.8	74.7	66.4
matched, same attributes, same level	**52.7**	**52.5**	**52.6**	**51.3**	**57.3**	**52.7**

Table A4. Precision (P) , recall (R) , and F1 scores for BN corpus with different criteria for matching reference and hypothesis relations (scorer's unweighted results shown in bold).

Criteria for Being Found	ST P	ST R	ST F1	RB P	RB R	RB F1
matched	84.1	45.8	59.3	77.3	55.8	64.8
matched, same type/subtype	69.2	37.7	48.8	62.0	44.8	52.0
matched, same attributes	60.3	32.9	42.5	52.3	37.7	43.8
matched, arg1&2 not missing (i.e. *mappable*)	57.3	31.2	40.4	57.5	41.5	48.2
matched, no missing ref. args	56.6	30.9	40.0	55.8	40.3	46.8
matched, same type/subtype, arg1&2 not missing	46.7	25.5	32.9	47.5	34.3	39.8
matched, same type/subtype, no missing ref. args	46.7	25.5	32.9	45.8	33.0	38.4
matched, same attributes, arg1&2 not missing	41.4	22.6	29.2	42.0	30.3	35.2
matched, same attributes, no missing ref. args	**41.4**	**22.6**	**29.2**	**41.3**	**29.8**	**34.6**

Table A5. Precision (P) , recall (R) , and F1 scores for CTS corpus with different criteria for matching reference and hypothesis entities (scorer's unweighted results shown in bold).

Criteria for Being Found	ST P	ST R	ST F1	RB P	RB R	RB F1
matched	87.6	67.4	76.2	79.1	69.1	73.8
matched, same type/subtype	74.8	57.6	65.1	69.1	60.4	64.5
matched, same attributes	40.9	31.5	35.6	42.1	36.8	39.3
matched, same level	79.6	61.2	69.2	77.2	67.4	72.0
matched, same type/subtype, same level	68.6	52.8	59.7	68.2	59.6	63.6
matched, same attributes, same level	**39.8**	**30.6**	**34.6**	**41.2**	**36.0**	**38.4**

Table A6. Precision (P) , recall (R) , and F1 scores for CTS corpus with different criteria for matching reference and hypothesis relations (scorer's unweighted results shown in bold).

Criteria for Being Found	ST P	ST R	ST F1	RB P	RB R	RB F1
matched	81.2	43.7	56.8	67.9	45.6	54.5
matched, same type/subtype	69.4	37.3	48.6	54.7	36.7	43.9
matched, same attributes	58.8	31.6	41.2	48.1	32.2	38.6
matched, arg1&2 not missing (i.e. *mappable*)	67.1	36.1	46.9	45.3	30.4	36.4
matched, no missing ref. args	65.9	35.4	46.1	45.3	30.4	36.4
matched, same type/subtype, arg1&2 not missing	57.6	31.9	40.3	38.7	25.9	31.1
matched, same type/subtype, no missing ref. args	56.5	30.4	39.5	38.7	25.9	31.1
matched, same attributes, arg1&2 not missing	50.6	27.2	35.4	34.9	23.4	28.0
matched, same attributes, no missing ref. args	**50.6**	**27.2**	**35.4**	**34.9**	**23.4**	**28.0**

Table A7. Precision (P) , recall (R) , and F1 scores for NW corpus with different criteria for matching reference and hypothesis entities (scorer's unweighted results shown in bold).

Criteria for Being Found	ST P	ST R	ST F1	RB P	RB R	RB F1
matched	78.6	84.5	81.4	77.8	87.6	82.4
matched, same type/subtype	63.3	67.9	65.5	66.3	74.7	70.3
matched, same attributes	51.4	55.2	53.2	53.7	60.5	56.9
matched, same level	75.1	80.7	77.8	74.3	83.7	78.7
matched, same type/subtype, same level	61.4	66.0	63.6	64.2	72.3	68.0
matched, same attributes, same level	**49.8**	**53.5**	**51.6**	**51.9**	**58.5**	**55.0**

Table A8. Precision (P) , recall (R) , and F1 scores for NW corpus with different criteria for matching reference and hypothesis relations (scorer's unweighted results shown in bold).

Criteria for Being Found	ST P	ST R	ST F1	RB P	RB R	RB F1
matched	79.4	50.4	61.7	72.4	53.2	61.4
matched, same type/subtype	65.2	41.4	50.7	58.4	42.9	49.5
matched, same attributes	53.9	34.3	41.9	47.1	34.6	39.9
matched, arg1&2 not missing (i.e. *mappable*)	50.7	32.2	39.4	51.1	37.6	43.3
matched, no missing ref. args	49.0	31.1	38.1	48.9	35.9	41.4
matched, same type/subtype, arg1&2 not missing	42.0	26.7	32.7	41.9	30.8	35.5
matched, same type/subtype, no missing ref. args	40.9	26.0	31.8	39.8	29.3	33.8
matched, same attributes, arg1&2 not missing	35.4	22.5	27.5	33.8	24.9	28.7
matched, same attributes, no missing ref. args	**34.8**	**22.1**	**27.0**	**32.8**	**24.1**	**27.8**

Table A9. Precision (P) , recall (R) , and F1 scores for UN corpus with different criteria for matching reference and hypothesis entities (scorer's unweighted results shown in bold).

Criteria for Being Found	ST P	ST R	ST F1	RB P	RB R	RB F1
matched	73.6	74.2	73.9	73.5	73.9	73.7
matched, same type/subtype	61.5	62.0	61.8	64.5	64.9	64.7
matched, same attributes	41.5	41.9	41.7	44.7	45.0	44.8
matched, same level	67.9	68.5	68.2	69.2	69.5	69.3
matched, same type/subtype, same level	58.2	58.7	58.4	61.7	62.0	61.9
matched, same attributes, same level	**39.7**	**40.1**	**39.9**	**42.7**	**42.9**	**42.8**

Table A10. Precision (P) , recall (R) , and F1 scores for UN corpus with different criteria for matching reference and hypothesis relations (scorer's unweighted results shown in bold).

Criteria for Being Found	ST P	ST R	ST F1	RB P	RB R	RB F1
matched	82.3	37.2	51.3	76.7	40.9	53.3
matched, same type/subtype	75.8	34.3	47.2	60.3	32.1	41.9
matched, same attributes	64.5	29.2	40.2	53.4	28.5	37.1
matched, arg1&2 not missing (i.e. *mappable*)	53.2	24.1	33.2	50.7	27.0	35.2
matched, no missing ref. args	51.6	23.4	32.2	50.7	27.0	35.2
matched, same type/subtype, arg1&2 not missing	51.6	23.4	32.2	39.7	21.2	27.6
matched, same type/subtype, no missing ref. args	50.0	22.6	31.2	39.7	21.2	27.6
matched, same attributes, arg1&2 not missing	46.8	21.2	29.1	35.6	19.0	24.8
matched, same attributes, no missing ref. args	**45.2**	**20.4**	**28.1**	**35.6**	**19.0**	**24.8**

Table A11. Precision (P) , recall (R) , and F1 scores for WL corpus with different criteria for matching reference and hypothesis entities (scorer's unweighted results shown in bold).

Criteria for Being Found	ST P	ST R	ST F1	RB P	RB R	RB F1
matched	75.3	81.5	78.3	75.5	84.4	79.7
matched, same type/subtype	59.4	64.3	61.8	62.9	70.3	66.4
matched, same attributes	44.1	47.7	45.8	46.2	51.7	48.8
matched, same level	67.8	73.4	70.5	72.2	80.7	76.2
matched, same type/subtype, same level	56.1	60.8	58.4	59.9	67.0	63.3
matched, same attributes, same level	**42.5**	**46.1**	**44.2**	**44.0**	**49.2**	**46.4**

Table A12. Precision (P) , recall (R) , and F1 scores for WL corpus with different criteria for matching reference and hypothesis relations (scorer's unweighted results shown in bold).

Criteria for Being Found	ST P	ST R	ST F1	RB P	RB R	RB F1
matched	72.2	44.0	54.7	67.2	49.1	56.7
matched, same type/subtype	63.9	39.0	48.4	54.3	39.6	45.8
matched, same attributes	50.5	30.8	38.3	43.1	31.4	36.4
matched, arg1&2 not missing (i.e. *mappable*)	42.3	25.8	32.0	47.4	34.6	40.0
matched, no missing ref. args	41.2	25.2	31.3	44.8	32.7	37.8
matched, same type/subtype, arg1&2 not missing	36.1	22.0	27.3	37.1	27.0	31.3
matched, same type/subtype, no missing ref. args	35.1	21.4	26.6	34.5	25.2	29.1
matched, same attributes, arg1&2 not missing	28.9	17.6	21.9	30.2	22.0	25.5
matched, same attributes, no missing ref. args	**28.9**	**17.6**	**21.9**	**30.2**	**22.0**	**25.5**

Table A13. Frequency of combinations of errors for ST. Certain conditions can occur that require correction using Table 10.

Subject Error	Object Error	Relation Type Error	Entity Type Error	BC	BN	CTS	NW	UN	WL
NA	NA	NA	NA	0.233	0.159	0.188	0.206	0.177	0.278
NA	NA	NA	NA	0.495	0.493	0.565	0.478	0.484	0.515
Pull-Down	NA	NA	NA	0.010	0.010	0.012	0.026	0.016	0.031
Copy-Paste	NA	NA	NA	0.019	0.040	0.035	0.023	0.065	0.031
NA	Pull-Down	NA	NA	0.039	0.007	0.024	0.023	0.032	0.010
NA	Copy-Paste	NA	NA	0.000	0.070	0.024	0.061	0.097	0.021
Pull-Down	Pull-Down	NA	NA	0.000	0.003	0.000	0.003	0.016	0.000
Copy-Paste	Pull-Down	NA	NA	0.000	0.000	0.012	0.000	0.016	0.000
Pull-Down	Copy-Paste	NA	NA	0.000	0.003	0.000	0.006	0.000	0.000
Copy-Paste	Copy-Paste	NA	NA	0.010	0.013	0.012	0.003	0.016	0.010
NA	NA	Pull-Down	NA	0.078	0.060	0.082	0.070	0.032	0.062
Pull-Down	NA	Pull-Down	NA	0.019	0.000	0.000	0.012	0.000	0.000
Copy-Paste	NA	Pull-Down	NA	0.010	0.013	0.000	0.000	0.000	0.000
NA	Pull-Down	Pull-Down	NA	0.010	0.000	0.012	0.009	0.000	0.000
NA	Copy-Paste	Pull-Down	NA	0.000	0.026	0.000	0.003	0.000	0.000
Pull-Down	Pull-Down	Pull-Down	NA	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Pull-Down	Pull-Down	NA	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	Copy-Paste	Pull-Down	NA	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Copy-Paste	Pull-Down	NA	0.000	0.000	0.000	0.000	0.000	0.000
NA	NA	NA	Pull-Down	0.029	0.017	0.012	0.017	0.000	0.010
Pull-Down	NA	NA	Pull-Down	0.000	0.003	0.000	0.000	0.000	0.010
Copy-Paste	NA	NA	Pull-Down	0.010	0.013	0.000	0.000	0.000	0.000
NA	Pull-Down	NA	Pull-Down	0.000	0.010	0.000	0.006	0.000	0.000
NA	Copy-Paste	NA	Pull-Down	0.000	0.003	0.000	0.000	0.000	0.000
Pull-Down	Pull-Down	NA	Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Pull-Down	NA	Pull-Down	0.000	0.000	0.000	0.003	0.016	0.000
Pull-Down	Copy-Paste	NA	Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Copy-Paste	NA	Pull-Down	0.000	0.003	0.000	0.003	0.000	0.000
NA	NA	Pull-Down	Pull-Down	0.019	0.020	0.024	0.014	0.032	0.000
Pull-Down	NA	Pull-Down	Pull-Down	0.000	0.000	0.000	0.003	0.000	0.000
Copy-Paste	NA	Pull-Down	Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
NA	Pull-Down	Pull-Down	Pull-Down	0.010	0.000	0.000	0.000	0.000	0.000
NA	Copy-Paste	Pull-Down	Pull-Down	0.000	0.010	0.000	0.006	0.000	0.010
Pull-Down	Pull-Down	Pull-Down	Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Pull-Down	Pull-Down	Pull-Down	0.000	0.003	0.000	0.000	0.000	0.000
Pull-Down	Copy-Paste	Pull-Down	Pull-Down	0.000	0.003	0.000	0.000	0.000	0.000
Copy-Paste	Copy-Paste	Pull-Down	Pull-Down	0.000	0.007	0.000	0.012	0.000	0.000
NA	NA	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	NA	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	NA	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
NA	Pull-Down	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
NA	Copy-Paste	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	Pull-Down	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Pull-Down	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	Copy-Paste	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Copy-Paste	NA	2 Pull-Down	0.010	0.003	0.000	0.000	0.000	0.000
NA	NA	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	NA	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	NA	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
NA	Pull-Down	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.006	0.000	0.000
NA	Copy-Paste	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	Pull-Down	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Pull-Down	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	Copy-Paste	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Copy-Paste	Pull-Down	2 Pull-Down	0.000	0.007	0.000	0.009	0.000	0.010

Table A14. Frequency of combinations of errors for RB. Certain conditions can occur requiring correction using Table 10.

Subject Error	Object Error	Relation Type Error	Entity Type Error	BC	BN	CTS	NW	UN	WL
NA	NA	NA	NA	0.331	0.228	0.321	0.276	0.233	0.328
NA	NA	NA	NA	0.271	0.430	0.321	0.398	0.329	0.405
Pull-Down	NA	NA	NA	0.008	0.008	0.047	0.020	0.027	0.026
Copy-Paste	NA	NA	NA	0.075	0.038	0.038	0.030	0.055	0.009
NA	Pull-Down	NA	NA	0.008	0.008	0.019	0.025	0.014	0.009
NA	Copy-Paste	NA	NA	0.023	0.058	0.009	0.035	0.068	0.017
Pull-Down	Pull-Down	NA	NA	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Pull-Down	NA	NA	0.008	0.000	0.000	0.000	0.014	0.000
Pull-Down	Copy-Paste	NA	NA	0.000	0.003	0.000	0.000	0.027	0.000
Copy-Paste	Copy-Paste	NA	NA	0.053	0.018	0.094	0.030	0.055	0.009
NA	NA	Pull-Down	NA	0.068	0.070	0.075	0.068	0.082	0.043
Pull-Down	NA	Pull-Down	NA	0.000	0.003	0.009	0.000	0.014	0.000
Copy-Paste	NA	Pull-Down	NA	0.023	0.005	0.000	0.005	0.014	0.017
NA	Pull-Down	Pull-Down	NA	0.000	0.000	0.000	0.000	0.000	0.000
NA	Copy-Paste	Pull-Down	NA	0.008	0.005	0.000	0.008	0.000	0.000
Pull-Down	Pull-Down	Pull-Down	NA	0.000	0.000	0.009	0.000	0.000	0.000
Copy-Paste	Pull-Down	Pull-Down	NA	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	Copy-Paste	Pull-Down	NA	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Copy-Paste	Pull-Down	NA	0.000	0.000	0.000	0.000	0.000	0.017
NA	NA	NA	Pull-Down	0.023	0.030	0.009	0.030	0.000	0.043
Pull-Down	NA	NA	Pull-Down	0.000	0.008	0.000	0.000	0.000	0.017
Copy-Paste	NA	NA	Pull-Down	0.008	0.005	0.009	0.003	0.000	0.000
NA	Pull-Down	NA	Pull-Down	0.000	0.008	0.000	0.003	0.000	0.000
NA	Copy-Paste	NA	Pull-Down	0.000	0.005	0.000	0.008	0.000	0.009
Pull-Down	Pull-Down	NA	Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Pull-Down	NA	Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	Copy-Paste	NA	Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Copy-Paste	NA	Pull-Down	0.008	0.003	0.000	0.003	0.000	0.000
NA	NA	Pull-Down	Pull-Down	0.015	0.030	0.038	0.033	0.027	0.026
Pull-Down	NA	Pull-Down	Pull-Down	0.000	0.003	0.000	0.000	0.000	0.000
Copy-Paste	NA	Pull-Down	Pull-Down	0.008	0.005	0.000	0.000	0.000	0.009
NA	Pull-Down	Pull-Down	Pull-Down	0.008	0.008	0.000	0.000	0.014	0.000
NA	Copy-Paste	Pull-Down	Pull-Down	0.023	0.015	0.000	0.010	0.000	0.000
Pull-Down	Pull-Down	Pull-Down	Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Pull-Down	Pull-Down	Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	Copy-Paste	Pull-Down	Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Copy-Paste	Pull-Down	Pull-Down	0.038	0.005	0.000	0.010	0.000	0.000
NA	NA	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	NA	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	NA	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
NA	Pull-Down	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
NA	Copy-Paste	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	Pull-Down	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Pull-Down	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	Copy-Paste	NA	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Copy-Paste	NA	2 Pull-Down	0.000	0.003	0.000	0.000	0.014	0.000
NA	NA	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	NA	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	NA	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
NA	Pull-Down	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
NA	Copy-Paste	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Pull-Down	Pull-Down	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Pull-Down	Pull-Down	2 Pull-Down	0.000	0.003	0.000	0.000	0.000	0.000
Pull-Down	Copy-Paste	Pull-Down	2 Pull-Down	0.000	0.000	0.000	0.000	0.000	0.000
Copy-Paste	Copy-Paste	Pull-Down	2 Pull-Down	0.000	0.003	0.000	0.008	0.014	0.017