HLT-NAACL 2004 Workshop: Biolink 2004,
Linking Biological Literature, Ontologies and Databases, pp. 1-8.
Association for Computational Linguistics.

A resource for constructing customized test suites for molecular biology entity identification systems

K. Bretonnel Cohen
Center for Computational Pharmacology,
University of Colorado School of Medicine
kevin.cohen@uchsc.edu

Lorraine Tanabe
National Center for Biotechnology
Information, NLM, NIH
tanabe@ncbi.nlm.nih.gov

Shuhei Kinoshita
Fujitsu Ltd. Bio-IT Lab, and Center for
Computational Pharmacology, University of
Colorado School of Medicine
shuhei.kinoshita@uchsc.edu

Lawrence Hunter
Center for Computational Pharmacology,
University of Colorado School of Medicine
larry.hunter@uchsc.edu

Abstract

This paper describes a data source and methodology for producing customized test suites for molecular biology entity identification systems. The data consists of: (a) a set of gene names and symbols classified by a taxonomy of features that are relevant to the performance of entity identification systems, and (b) a set of sentential environments into which names and symbols are inserted to create test data and the associated gold standard. We illustrate the utility of test sets producible by this methodology by applying it to five entity identification systems and describing the error patterns uncovered by it, and investigate relationships between performance on a customized test suite generated from this data and the performance of a system on two corpora.

1 Introduction

This paper describes a methodology and data for the testing of molecular biology entity identification (EI) systems by developers and end users. Molecular biology EI systems find names of genes and gene products in free text. Several years' publication history has established precision, recall, and F-score as the de facto standards for evaluating EI systems for molecular biology texts at the publication stage and in competitions like BioCreative (www.mitre.org/public/biocreative). These measures provide important indices of a system's overall output quality. What they do not provide is the detailed sort of information about system performance that is useful for the system developer who is attempting to assess the strengths and weaknesses of a work in progress, nor do they provide detailed information to the potential consumer who would like to compare two systems against each other. Hirschman and Mani (2003) point out that different evaluation methods are useful at different points in the software life-cycle. In particular, what they refer to as *feature-based* evaluation via test suites is useful at two points: in the development phase, and for acceptance testing. We describe here a methodology and a set of data for constructing customized feature-based test suites for EI in the molecular biology domain. The data consists of two sets. One is a set of names and symbols of *entities* as that term is most commonly understood in the molecular biology domain—genes and gene products. (Sophisticated ontologies such as GENIA (Ohta et al. 2002) include other kinds of entities relevant to molecular biology as well, such as *cell lines*.) The names and symbols exemplify a wide range of the features that characterize entities in this domain—case variation, presence or absence of numbers, presence or absence of hyphenation, etc. The other is a set of sentences that exemplify a range of *sentential contexts* in which the entities can appear, varying with respect to position of the entity in the sentence (initial, medial, or final), presence of keywords like *gene* and *protein*, tokenization issues, etc. Both the entities and the sentential contexts are classified in terms of a taxonomy of features that are relevant to this domain in particular and to natural language processing and EI in general. The methodology consists of generating customized test suites that address specific performance issues by combining sets of entities that have particular characteristics with sets of contexts that have particular characteristics. Logical combination of subsets of characteristics of entities and contexts allows the developer to assess the effect of specific characteristics on performance, and allows the user to assess performance of the system on types of inputs that are of

particular interest to them. For example, if the developer or end-user wants to assess the ability of a system to recognize gene symbols with a particular combination of letter case, hyphenation, and presence or absence of numerals, the data and associated code that we provide can be used to generate a test suite consisting of symbols with and without that combination of features in a variety of sentential contexts.

Inspiration for this work comes on the one hand from standard principles of software engineering and software testing, and on the other hand from descriptive linguistics (Harris 1951, Samarin 1967). In Hirschman and Mani's taxonomy of evaluation techniques, our methodology is referred to as *feature-based*, in that it is based on the principle of classifying the inputs to the system in terms of some set of features that are relevant to the application of interest. It is designed to provide the developer or user with detailed information about the performance of her EI system. We apply it to five molecular biology EI and information extraction systems: ABGene (Tanabe and Wilbur 2002a, Tanabe and Wilbur 2002b); KeX/PROPER (Fukuda et al. 1997); Yapex (Franzén et al. 2002); the stochastic POS tagging-based system described in Cohen et al. (in submission); and the entity identification component of Ono et al.'s information extraction system (Ono et al. 2001), and show how it gives detailed useful information about each that is not apparent from the standard metrics and that is not documented in the cited publications. (Since we are not interested in punishing system developers for graciously making their work available by pointing out their flaws, we do not refer to the various systems by name in the remainder of this paper.)

Software testing techniques can be grouped into structured (Beizer 1990), heuristic (Kaner et al. 2002), and random categories. Testing an EI system by running it on a corpus of texts and calculating precision, recall, and F-score for the results falls into the category of random testing. Random testing is a powerful technique, in that it is successful in finding bugs. When done for the purpose of evaluation, as distinct from testing (see Hirschman and Thompson 1997 for the distinction between the two, referred to there as *performance evaluation* and *diagnostic evaluation*), it also is widely accepted as the relevant index of performance for publication. However, its output lacks important information that is useful to a system developer (or consumer): it tells you how often the system failed, but not what it failed at; it tells you how often the system succeeds, but not where its strengths are.

For the developer or the user, a structured test suite offers a number of advantages in answering these sorts of questions. The utility of such test suites in general software testing is well-accepted. Oepen et al. (1998) lists a number of advantages of test suites vs. naturalistic corpora for testing natural language processing software in particular:

- **Control over test data:** test suites allow for "focussed and fine-grained diagnosis of system performance" (15). This is important to the developer who wants to know exactly what problems need to be fixed to improve performance, and to the end user who wants to know that performance is adequate on exactly the data that they are interested in.
- **Systematic coverage:** test suites can allow for systematic evaluation of variations in a particular feature of interest. For example, the developer might want to evaluate how performance varies as a function of name length, or case, or the presence or absence of hyphenation within gene symbols. The alternative to using a structured test suite is to use a corpus, and then search through it for the relevant inputs and hope that they are actually attested.
- **Control of redundancy:** while redundancy in a corpus is representative of actual redundancy in inputs, test suites allow for reduction of redundancy when it obscures the situation, or for increasing it when it is important to test handling of a feature whose importance is greater than its frequency in naturally occurring data. For example, names of genes that are similar to names of inherited diseases might make up only a small proportion of the gene names that occur in PubMed abstracts, but the user whose interests lie in curating OMIM might want to be able to assure herself that coverage of such names is adequate, beyond the level to which corpus data will allow.
- **Inclusion of negative data:** in the molecular biology domain, a test suite can allow for systematic evaluation of potential false positives.
- **Coherent annotation:** even the richest metadata is rarely adequate or exactly appropriate for exactly the questions that one wants to ask of a corpus. Generation of structured, feature-based test suites obviates the necessity for searching through corpora for the entities and contexts of interest, and allows instead the structuring of contexts and labeling of examples that is most useful to the developer.

The goal of this paper is to describe a methodology and publicly available data set for constructing customized and refinable test suites in the molecular biology domain quickly and easily. A crucial difference between similar work that simply documents a distributable test suite (e.g. Oepen (1998) and Volk (1998)) and the work reported in this paper is that we are distributing not a static test suite, but rather data for

generating test suites—data that is structured and classified in such a way as to allow software developers and end users to easily generate test suites that are customized to their own assessment needs and development questions. We build this methodology and data on basic principles of software engineering and of linguistic analysis. The first such principle involves making use of the software testing notion of the *catalogue*.

A *catalogue* is a list of test conditions, or qualities of particular test inputs (Marick 1997). It corresponds to the *features* of feature-based testing, discussed in Hirschman and Mani (2003) and to the *schedule* (Samarin 1967:108-112) of descriptive linguistic technique. For instance, a catalogue of test conditions for numbers might include:

- zero, non-zero, real, integer
- positive, negative, unsigned
- the smallest number representable in some data type, language, or operating system; smaller than the smallest number representable
- the largest number representable; larger than the largest number representable

Note that the catalogue includes both "clean" conditions and "dirty" ones. This approach to software testing has been highly successful, and indeed the best-selling book on software testing (Kaner et al. 1999) can fairly be described as a collection of catalogues of various types.

The contributions of descriptive linguistics include guiding our thinking about what the relevant features, conditions, or categories are for our domain of interest. In this domain, that will include the questions of what features may occur in names and what features may occur in sentences—particularly features in the one that might interact with features in the other. Descriptive linguistic methodology is described in detail in e.g. Harris (1951) and Samarin (1967); in the interests of brevity, we focus on the software engineering perspective here, but the thought process is very similar. The software engineering equivalent of the descriptive linguist's hypothesis is the *fault model* (Binder 1999)—an explicit hypothesis about a potential source of error based on "relationships and components of the system under test" (p. 1088). For instance, knowing that some EI systems make use of POS tag information, we might hypothesize that the presence of some parts of speech within a gene name might be mistaken for term boundaries (e.g. the *of* in *bag of marbles*, LocusID 43038). Catalogues are used to develop a set of test cases that satisfies the various qualities. (They can also be used post-hoc to group the inputs in a random test bed into equivalence classes, although a strong motivation for using them in the first place is to obviate this sort of search-based post-hoc analysis.) The size of the space of all possible test cases can be estimated from the Cartesian product of all catalogues; the art of software testing (and linguistic fieldwork) consisting, then, of selecting the highest-yielding subset of this often enormous space that can be run and evaluated in the time available for testing.

At least three kinds of catalogues are relevant to testing an EI system. They fall into one of two very broad categories: syntagmatic, having to do with combinatory properties, and paradigmatic, having to do with varieties of content. The three kinds of catalogues are:

1. A catalogue of environments in which gene names can appear. This is syntagmatic.
2. A catalogue of types of gene names. This is paradigmatic.
3. A catalogue of false positives. This is both syntagmatic and paradigmatic.

The catalogue of environments would include, for example, elements related to sentence position, such as sentence-initial, sentence-medial, and sentence-final; elements related to list position, such as a single gene name, a name in a comma-separated list, or a name in a conjoined noun phrase; and elements related to typographic context, such as location within parentheses (or not), having attached punctuation (e.g. a sentence-final period) (or not), etc. The catalogue of types of names would include, for example, names that are common English words (or not); names that are words versus "names" that are symbols; single-word versus multi-word names; and so on. The second category also includes typographic features of gene names, e.g. containing numbers (or not), consisting of all caps (or not), etc. We determined candidate features for inclusion in the catalogues through standard structuralist techniques such as examining public-domain databases containing information about genes, including FlyBase, LocusLink, and HUGO, and by examining corpora of scientific writing about genes, and also by the software engineering techniques of "common sense, experience, suspicion, analysis, [and] experiment" (Binder 1999). The catalogues then suggested the features by which we classified and varied the entities and sentences in the data.

General format of the data

The entities and sentences are distributed in XML format and are available at a supplemental web site (compbio.uchsc.edu/Hunter_lab/testing_ei). A plain-text version is also available. A representative entity is illustrated in Figure 1 below, and a representative sentence is illustrated in Figure 2. All data in the current version is restricted to the ASCII character set.

Test suite generation

Data sets are produced by selecting sets of entity features and sets of sentential context features and inserting the entities into slots in the sentences. This

can be accomplished with the user's own tools, or using applications available at the supplemental web site. The provided applications produce two files: a file containing raw data for use as test inputs, and a file containing the corresponding gold standard data marked up in an SGML-like format. For example, if the raw data file contains the sentence *ACOX2 polymorphisms may be correlated with an increased risk of larynx cancer*, then the gold standard file will contain the corresponding sentence *<gp>ACOX2</gp> polymorphisms may be correlated with an increased risk of larynx cancer*. Not all users will necessarily agree on what counts as the "right" gold standard—see Olsson et al. (2002) and the BioCreative site for some of the issues. Users can enforce their own notions of correctness by using our data as input to their own generation code, or by post-processing the output of our applications.

```
ID: 136
name_vs_symbol: n
length: 3
case: a
contains_a_numeral: y
contains_Arabic_numeral: y
Arabic_numeral_position: f
contains_Roman_numeral:
<several typographic features omitted>
contains_punctuation: 1
contains_hyphen: 1
contains_forward_slash:
<several punctuation-related features omitted>
contains_function_word:
function_word_position:
contains_past_participle: 1
past_participle_position: i
contains_present_participle:
present_participle_position:
source_authority: HGNC ID: 2681 "Approved Gene
Name" field
original_form_in_source:       death-associated
protein 6
data: death-associated protein 6
```

Figure 1 A representative entry from the entity data file. A number of null-valued features are omitted for brevity—see the full entry at the supplemental web site. The *data* field (last line of the figure) is what is output by the generation software.

```
ID: 25
type: tp
total_number_of_names: 1
list_context:
position: I
typographic_context:
appositive:
source_id: PMID: 14702106
source_type: title
original_form_in_source: Stat-3 is required
for pulmonary homeostasis during hyperoxia.
slots: <> is required for pulmonary
homeostasis during hyperoxia.
```

Figure 2 A representative entry from the sentences file. Features and values are explained in section 2.2 *Feature set for sentential contexts* below. The *slots* field (last line of the figure) shows where an entity would be inserted when generating test data.

2 The taxonomy of features for entities and sentential contexts

In this section we describe the feature sets for entities and sentences, and motivate the inclusion of each, where not obvious.

2.1 Feature set for entities

Conceptually, the features for describing name-inputs are separated into four categories: orthographic/typographic, morphosyntactic, source, and lexical.

- Orthographic/typographic features describe the presence or absence of features on the level of individual characters, for example the case of letters, the presence or absence of punctuation marks, and the presence or absence of numerals.
- Morphosyntactic features describe the presence or absence of features on the level of the morpheme or word, such as the presence or absence of participles, the presence or absence of genitives, and the presence or absence of function words.
- Source features are defined with reference to the source of an input. (It should be noted that in software engineering, as in Chomskyan theoretical linguistics, data need not be naturally-occurring to be useful; however, with the wealth of data available for gene names, there is no reason not to include naturalistic data, and knowing its source may be useful, e.g. in evaluating performance on FlyBase names, etc.) Source features include source type, e.g. literature, database, or invention; identifiers in a database; canonical form of the entity in the database; etc.
- Lexical features are defined with respect to the relationship between an input and some outside source of lexical information, for instance whether or not an input is or contains a common English word. This is also the place to indicate whether or not an input is present in a resource such as LocusLink, whether or not it is on a particular stoplist, whether it is in-vocabulary or out-of-vocabulary for a particular language model, etc.

The distinction between these three broad categories of features is not always clear-cut. For example, presence of numerals is an orthographic/typographic feature, and is also morphosyntactic when the numeral postmodifies a noun, e.g. in *heat shock protein 60*. Likewise, features may be redundant—for example, the presence of a Greek letter in the square-bracket- or curly-bracket-

enclosed formats, or the presence of an apostrophized genitive, are not independent of the presence of the associated punctuation marks. However, Boolean queries over the separate feature sets let them be manipulated and queried independently. So, entities with names like *A′* can be selected independently of names like *Parkinson's disease.*

2.1.1 Orthographic/typographic features

Length: Length is defined in characters for symbols and in whitespace-tokenized words for names.

Case: This feature is defined in terms of five possible values: all-upper-case, all-lower-case, upper-case-initial-only, each-word-upper-case-initial (e.g. *Pray For Elves*), and mixed. The fault model motivating this feature hypothesizes that taggers may rely on case to recognize entities and may fail on some combinations of cases with particular sentential positions. For example, one system performed well on gene symbols in general, except when the symbols are lower-case-initial and in sentence-initial position (e.g. *p100 is abundantly expressed in liver…* (PMID 1722209) and *bif displays strong genetic interaction with msn* (PMID 12467587).

Numeral-related features: A set of features encodes whether or not an entity contains a numeral, whether the numeral is Arabic or Roman, and the positions of numerals within the entity (initial, medial, or final). The motivation for this feature is the hypothesis that a system might be sensitive to the presence or absence of numerals in entities. One system failed when the entity was a name (vs. a symbol), it contained a number, and the number was in the right-most (vs. a medial) position in a word. It correctly tagged entities like *glucose 6 phosphate dehydrogenase* but missed the boundary on *<gp>alcohol dehydrogenase</gp> 6.* This pattern was specific to numbers—letters in the same position are handled correctly.

Punctuation-related features: A set of features includes whether an entity contains any punctuation, the count of punctuation marks, and which marks they are (hyphen, apostrophe, etc.). One system failed to recognize names (but typically not symbols) when they included hyphens. Another system had a very reliable pattern of failure involving apostrophes just in case they were in genitives.

Greek-letter-related features: These features encode whether or not an entity contains a Greek letter, the position of the letter, and the format of the letter. (This feature is an example of an orthographic feature which may be defined on a substring longer than a character, e.g. *beta*.) Two systems had problems recognizing gene names when they contained Greek letters in the PubMed Central format, i.e. *[beta]1 integrin.*

2.1.2 Morphosyntactic features

The most salient morphosyntactic feature is whether an entity is a name or a symbol. The fault model motivating this feature suggests that a system might perform differently depending on whether an input is a name or a symbol. The most extreme case of a system being sensitive to this feature was one system that performed very well on symbols but recognized no names whatsoever.

Features related to function words: a set of features encodes whether or not an entity contains a function word, the number of function words in the entity, and their positions—for instance, the facts: that *scott of the antarctic* (FlyBase ID FBgn0015538) contains two function words; that they are *of* and *the;* and that they are medial to the string. This feature is motivated by two fault models. One posits that a system might apply a stoplist to its input and that processing of function words might therefore halt at an early stage. The other posits that a system might employ shallow parsing to find boundaries of entities and that the shallow parser might insert boundaries at the locations of function words, causing some words to be omitted from the entity. One system always had partial hits on names that were multi-word unless each word in it was upper-case-initial, or there was an alphanumeric postmodifier (i.e. a numeral, upper-cased singleton letter, or Greek letter) at the right edge.

Features related to inflectional morphology: a set of features encodes whether or not an entity contains nominal number or genitive morphology or verbal participial morphology, and the positions of the words in the entity that contain those morphemes, for instance the facts that *apoptosis antagonizing transcription factor* (HUGO ID 19235) contains a present participle and that the word that contains it is medial to the string.

Features related to parts of speech: Future development of the data will include features encoding the parts of speech present in names.

2.1.3 Source features

Source or authority: This feature encodes the source of or authority cited for an entity. For many of the entries in the current data, it is an identifier from some database. For others, it is a website (e.g. www.flynome.org). Other possible values include the PMID of a document in which it was observed.

Original form in source: Where there is a source for the entity or for some canonical form of the entity, the original form is given. This is *not* equivalent to the "official" form, but rather is the exact form in which the entity occurs; it may even contain typographic errors (e.g. the extraneous space in *nima –related kinase,* LocusID 189769 (reported to the NCBI service desk).

2.1.4 Lexical features

These might be better called *lexicographic features.* They can be encoded impressionistically, or can be defined with respect to an external source, such as WordNet, the UMLS, or other lexical resources. They may also be useful for encoding strictly local information, such as whether or not a gene was attested in training data or whether it is present in a particular language model or other local resource. These features are allowed in the taxonomy but are not implemented in the current data. Our own use of the entity data suggests that it should be, especially encoding of whether or not names include common English words. (The presence of function words is already encoded.)

2.2 Feature set for sentential contexts

In many ways, this data is much harder to build and classify than the names data, for at least two reasons. Many more features interact with each other, and as soon as a sentence contains more than one gene name, it contains more than one environment, and the number of features for the sentence as a whole is multiplied, as are the interactions between them. For this reason, we have focussed our attention so far on sentences containing only a single gene name, although the current version of the data does include a number of multi-name sentences.

2.2.1 Positivity

The fundamental distinction in the feature set for sentences has to do with whether the sentence is intended to provide an environment in which gene names actually appear, or whether it is intended to provide a non-trivial opportunity for false positives.

True positive sentences contain some *slot* in which entities from the names data can be inserted, e.g. <> *polymorphisms may be correlated with an increased risk of larynx cancer* or <> *interacts with* <> *and* <> *in the two-hybrid system.*

False positive sentences contain one or more tokens that are deliberately intended to pose challenging opportunities for false positives. Certainly any sentence which does not consist all and only of a single gene name contains opportunities for false positives, but not all potential false positives are created equal. We include in the data set sentences that contain tokens with orthographic and typographic characteristics that mimic the patterns commonly seen in gene names and symbols, e.g. *The aim of the present study is to evaluate the impact on QoL…* where *QoL* is an abbreviation for *quality of life*. We also include sentences that contain "keywords" that may often be associated with genes, such as *gene, protein, mutant, expression,* etc., e.g. *Demonstration of antifreeze protein activity in Antarctic lake bacteria.*

2.2.2 Features for TP sentences

Number and positional features encode the total number of slots in the sentence, and their positions. The value for the *position* feature is a list whose values range over initial, medial, and final. For example, the sentence <> *interacts with* <> *and* <> *in the two-hybrid system* has the value *I,M* (initial and medial) for the *position* feature.

Typographic context features encode issues related to tokenization, specifically related to punctuation, for example if a slot has punctuation on the left or right edge, and the identity of the punctuation marks.

List context features encode data about position in lists. These include the type of list (coordination, asyndetic coordination, or complex coordination).

The **appositive feature** is for the special case of appositioned symbols or abbreviations and their full names or definitions, e.g. *The Arabidopsis INNER NO OUTER (INO) gene is essential for formation and…* For the systems that we have tested with it, it has not revealed problems that are independent of the typographic context. However, we expect it to be of future use in testing systems for abbreviation expansion in this domain.

Source features encode the identification and type of the source for the sentence and its original form in the source. The source identifier is often a PubMed ID. It bears pointing out again that there is no a priori reason to use sentences with any naturally-occurring "source" at all, as opposed to the products of the software engineer's imagination. Our primary rationale for using naturalistic sources at all for the sentence data has more to do with convincing the user that some of the combinations of entity features and sentential features that we claim to be worth generating actually do occur. For instance, it might seem counterintuitive that gene symbols or names would ever occur lower-case-initial in sentence initial position, but in fact we found many instances of this phenomenon; or that a multi-word gene name would occur in text in all upper-case letters, but see the *INNER NO OUTER* example above.

Syntactic features encode the characteristics of the local environment. Some are very lexical, such as: whether the following word is a keyword; whether the preceding word is a species name. Others are more abstract, such as whether the preceding word is an article; whether the preceding word is an adjective; whether the preceding word is a conjunction; whether the preceding word is a preposition. Interactions with the list context features are complex. The fault model motivating these features hypothesizes that POS context and the presence of keywords might affect a system's judgments about the presence and boundaries of names.

2.2.3 Features for FP sentences

Most features for FP sentences encode the characteristics that give the contents of the sentence their FP potential. The *keyword* feature is a list of keywords present in the sentence, e.g. *gene, protein, expression*, etc. The *typographic features* feature encodes whether or not the FP potential comes from orthographic or typographic features of some token in the sentence, such as mixed case, containing hyphens and a number, etc. The *morphological features* feature encodes whether or not the FP potential comes from apparent morphology, such as words that end with *ase* or *in*.

3 Testing the relationship between predictions from performance on a test suite and performance on a corpus

Precision and recall on data in a structured test suite should not be expected to predict precision and recall on a corpus, since there is no relation between the prevalence of features in the test suite and prevalence of features in the corpus. However, we hypothesized that performance on an equivalence class of inputs in a test suite might predict performance on the same equivalence class in a corpus. To test this hypothesis, we ran a number of test suites through one of the systems and analyzed the results, looking for patterns of errors. The test suites were very simple, varying only entity length, case, hyphenation, and sentence position. Then we ran two corpora through the same system and examined the output for the actual corpora to see if the predictions based on the system's behavior on the test suite actually described performance on similar entities in the corpora.

One corpus, which we refer to as PMC (since it was sampled from PubMed Central), consists of 2417 sentences sampled randomly from a set of 1000 full-text articles. This corpus contains 3491 entities. It is described in Tanabe and Wilbur (2002b). The second corpus was distributed as training data for the BioCreative competition. It consists of 10,000 sentences containing 11,851 entities and is described in detail at www.mitre.org/public/biocreative. Each corpus is annotated for entities.

The predictions based on the system's performance on the test suite data were:

1. The system will have low recall on entities that have numerals in initial position, followed by a dash, e.g. *825-Oak*, *12-LOX*, and *18-wheeler* (`/^\d+-/` in Perl).
2. The system will have low recall on names that contain stopwords, such as *Pray For Elves* and *ken and barbie*.
3. The system will have low recall on sentence-medial terms that begin with a capital letter, such as *Always Early*.
4. The system will have low recall on three-character-long symbols.
5. The system will have good recall on (long) names that end with numerals.

We then examined the system's true positive, false positive, and false negative outputs from the two corpora for outputs that belonged to the equivalence classes in 1-5. Table 1 shows the results.

	BioCreative				
	TP	FP	FN	P	R
1	12	57	17	.17	.41
2	0	1	38	0.0	0.0
4	556	278	512	.67	.52
5	284	251	72	.53	.80
	PubMed Central				
	TP	FP	FN	P	R
1	8	10	0	.44	1.0
2	1	0	2	1.0	.33
4	163	64	188	.72	.46
5	108	54	46	.67	.70

Table 1 Performance on two corpora for the predictable categories Numbers in the far left column refer to the predictions listed above. Overall performance on the corpora was: BioCreative P = .65, R = .68, and PMC P = .71, R = .62.

For equivalence classes 1, 2, and 4, the predictions mostly held. Low recall was predicted, and actual recall was .41, 0.0, .52, 1.0 (the one anomaly), .33, and .46 for these classes of names, versus overall recall of .68 on the BioCreative corpus and .62 on the PMC corpus. The prediction held for equivalence class 5, as well; good recall was predicted, and actual recall was .80 and .70—higher than the overall recalls for the two corpora. The third prediction could not be evaluated due to the normalization of case in the gold standards. These results suggest that a test suite can be a good predictor of performance on entities with particular typographic characteristics.

4 Conclusion

We do not advocate using this approach to replace the quantitative evaluation of EI systems by precision, recall, and F-measure. Arguably, overall performance on real corpora is the best evaluation metric for entity identification, in which case the standard metrics are well-suited to the task. However, at specific points in the software lifecycle, viz. during development and at the time of acceptance testing, the standard metrics do not provide the right kind of information. We can,

however, get at this information if we bear in mind two things:

1. Entity identification systems are software, and as such can be assessed by standard software testing techniques.
2. Entity identification systems are in some sense instantiations of hypotheses about linguistic structure, and as such can be assessed by standard linguistic "field methods."

This paper describes a methodology and a data set for utilizing the principles of software engineering and linguistic analysis to generate test suites that answer the right kinds of questions for developers and for end users. Readers are invited to contribute their own data.

Acknowledgments

The authors gratefully acknowledge support for this work from NIH/NIAAA grant U01-AA13524-02; comments from Andrew E. Dolbey on an earlier version of this work; Philip V. Ogren for help with stochastic-POS-tagging-based system; the Center for Computational Pharmacology NLP reading group and the anonymous reviewers for insightful comments on the current version; and Fukuda et al., Ono et al., and Franzén et al. for generously making their systems publicly available.

References

Beizer, Boris (1990). *Software testing techniques, 2nd ed.* Van Nostrand Reinhold.

Binder, Robert V. (1999). *Testing object-oriented systems: models, patterns, and tools.* Addison-Wesley.

Cohen, K. Bretonnel; Philip V. Ogren; Shuhei Kinoshita; and Lawrence Hunter (in submission). Entity identification in the molecular biology domain with a stochastic POS tagger. ISMB 2004.

Cole, Ronald; Joseph Mariani; Hans Uszkoreit; Annie Zaenen; and Victor Zue (1997). *Survey of the state of the art in human language technology.* Cambridge University Press.

Franzén, Kristofer; Gunnar Eriksson; Fredrik Olsson; Lars Asker; Per Lidén; and Joakim Cöster (2002). Protein names and how to find them. *International Journal of Medical Informatics* 67(1-3):49-61.

Fukuda, K.; T. Tsunoda; A. Tamura; and T. Takagi (1997). Toward information extraction: identifying protein names from biological papers. *Pacific Symposium on Biocomputing 1998*, pp. 705-716.

Harris, Zellig S. (1951). *Methods in structural linguistics.* University of Chicago Press.

Hirschman, Lynette; and Inderjeet Mani (2003). Evaluation. In Mitkov (2003), pp. 415-429.

Hirschman, Lynette; and Henry S. Thompson (1997). Overview of evaluation in speech and natural language processing. In Cole et al. (1997), pp. 409-414.

Kaner, Cem; Hung Quoc Nguyen; and Jack Falk (1999). *Testing computer software, 2nd ed.* John Wiley & Sons.

Kaner, Cem; James Bach; and Bret Pettichord (2002). *Lessons learned in software testing: a context-driven approach.* John Wiley & Sons.

Marick, Brian (1997). *The craft of software testing: subsystem testing including object-based and object-oriented testing.* Prentice Hall.

Mitkov, Ruslan (2003). *The Oxford Handbook of Computational Linguistics.* Oxford University Press.

Nerbonne, John (1998). *Linguistic Databases.* CSLI Publications.

Ohta, Tomoko; Yuka Tateisi; Jin-Dong Kim; Hideki Mima; and Jun-ichi Tsujii (2002). The GENIA corpus: an annotated corpus in molecular biology. *Proceedings of the Human Language Technology Conference.*

Oepen, Stephan; Klaus Netter; and Judith Klein (1998). TSNLP – Test Suites for Natural Language Processing. In Nerbonne (1998), pp. 13-36.

Olsson, Fredrik; Gunnar Eriksson; Kristofer Franzén; Lars Asker; and Per Lidén (2002). Notions of correctness when evaluating protein name taggers. *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002),* Taipei, Taiwan.

Ono, Toshihide; Haretsugu Hishigaki; Akira Tanigami; and Toshihisa Takagi (2001). Automated extraction of information on protein-protein interactions from the biological literature. *Bioinformatics* 17(2):155-161.

Samarin, William J. (1967). *Field linguistics: a guide to linguistic field work.* Irvington.

Tanabe, Lorraine; and W. John Wilbur (2002a). Tagging gene and protein names in biomedical text. *Bioinformatics* 18(8):1124-1132.

Tanabe, Lorraine; and W. John Wilbur (2002b). Tagging gene and protein names in full text articles. *Proceedings of the workshop on natural language processing in the biomedical domain*, pp. 9-13. Association for Computational Linguistics.

Volk, Martin (1998). Markup of a test suite with SGML. In Nerbonne (1998), pp. 59-76.