Notions of Correctness when Evaluating Protein Name Taggers

Fredrik Olsson, Gunnar Eriksson, Kristofer Franzén
Swedish Institute of Computer Science
Box 1263, SE-164 29 Kista, Sweden

Lars Asker, Per Lidén
Virtual Genetics Laboratory AB
SE-171 77 Stockholm, Sweden

Abstract

This paper introduces four different notions of correctness to be used when measuring the performance of protein name taggers, each of which reflects certain characteristics of the tagger under evaluation. The discussion regarding the different notions is centered around the evaluation of two protein name taggers; Yapex, developed by the authors, and KeX developed by Fukuda et al. (1998). For the purpose of illustrating the difference between the ways of evaluation, both taggers are applied to a test corpus of 101 MEDLINE abstracts in which all occurrences of protein names have been marked up by domain experts.

1 Introduction

The roles and functions of proteins are important study objects in many areas of the life sciences, as well as in the pharmaceutical industry. In view of the vast amount of scientific text produced in these areas, it is important to develop tools and methods for automatic structuring and extraction of the information found therein.

The detection and categorization of named entities, such as names of people, organisations and places, in classical MUC-style information extraction tasks, e.g., (Borthwick et al., 1998) might be regarded a solved problem. But names of proteins present a different challenge because of their variant structural characteristics, their resemblance to regular noun phrases and their oftentimes unclear position on the continuum between proper names and terminology. They share these characteristics with other biological substances and probably with many other kinds of *named terminology* as well.

One common reason for developing methods for automatic detection of protein names in text has been the desire to build systems for automatic extraction of interactions between proteins, e.g., (Blaschke et al., 1999; Thomas et al., 2000). However, the detection of protein names is useful in itself. In our case, the first application at hand is a browsing support system, which links protein names in scientific text to entries in SWISS-PROT (Bairoch and Apweiler, 2000), which is an annotated protein sequence database.

Previous attempts at identifying protein names in text can be divided into systems using machine learning techniques, e.g., (Nobata et al., 1999; Collier et al., 2000), and systems based on hand-written rules, e.g., (Fukuda et al., 1998; Humphreys et al., 2000). The advantage of using machine learning techniques is that such a system is relatively easy to tune to new domains, provided that tagged training data exists. A rule based system, on the other hand, requires a lot of human analysis and labor, but results in a transparent system which is easier to support, adjust and expand. In Yapex, we utilise an off-the-shelf syntactic parser to improve the performance of a rule-based system.

Work on evaluation of protein name taggers seldom clearly specify what notion of correctness has been used when evaluating the systems, with the exception of de Bruijn and Martin (2000), who present figures on *undertagging* and *overtagging*, as well as *type* and *token* matches. Correspondingly, the definition of what should be regarded a protein name is often implicit.

2 Protein Names

Despite the lack of common standards and fixed nomenclatures, protein names exhibit several regularities that can be exploited in order to identify previously unseen instances. Primarily, protein names are almost always descrip-

tive in some way. Protein characteristics such as function (e.g. *growth hormone*), localization or cellular origin (such as *HIV-1 envelope glycoprotein gp120*), physical properties (*salivary acidic protein-1*), similarities to other proteins (*Rho-like protein*) are commonly reflected in the name. Names are also constructed using a combination or abbreviation of the above. As can be noted from the examples, protein names often consist of multiple words.

It needs to be said that the definition of what should be considered a protein name is not self-evident and that it can be varied to a certain extent. In this study, we define a protein name semantically as something that denotes a single biological entity composed of one or more amino acid chains. Protein fragments or protein families are not included in this definition.

In addition to the semantic definition above, from a text structural point of view, we define a protein name as a sequence of words denoting a specific, individual protein entity. Furthermore, we also include some, more indirect, references to individual protein entities into the protein name definition, (e.g. `<prot>`*importin beta1*`</prot>` *derivatives*). The definition excludes non-specific reference to individuals (*transcription factor*, *a 89 kD protein*). It also excludes most reference to groups or classes of proteins (*protein kinases*, *globulins*), though phrases denoting small groups of nearly identical proteins are included (*eukaryotic RhoA-binding kinases*).

Finally, the definition of a protein name excludes anaphoric references to proteins (*this protein*).

3 Yet another protein name extractor — Yapex

Arguably, building information extraction systems always involves decisions regarding how to balance recall and precision; depending on the application, one may want to focus on one or the other. Yapex initially strives for high recall with the consequence of poor precision. Later modules in the pipelined system use filtering techniques and syntactic information to boost precision, and a local dynamic dictionary is eventually applied to increase recall.

The Yapex algorithm can be described as consisting of the seven steps described below: The first four steps are concerned with the lexical analysis of singel word tokens, and the first two of these are implementations of some of the heuristic steps in the algorithm described by Fukuda et al. (1998) from which the terminology of these steps is borrowed. Steps five and six are concerned with the syntactic analysis of noun phrases and of the lexical categories derived in the previous steps, and the final step utilizes the syntactic information gathered to identify new single- or multi-word protein names.

The Yapex system is available for testing at `http://www.sics.se/humle/projects/prothalt/`.

3.1 Lexical analysis of feature terms

Feature terms are words, e.g., *receptor* and *enzyme*, that describe the function or characteristics of a protein. These words often occur in or nearby a protein name and can be used as indicators of the presence of such a name. The analysis discriminates between internal and external feature terms, internal terms being words that belong to the name like *protein*, *particle*, and *receptor*. External feature terms are words – e.g. *peptide*, *domain*, and *terminal* – that acts as indicators of a protein name but, most often, does not constitute a part of the name itself, according to our protein name definition. Among the internal feature terms we treat strong terms separatly. These terms (*factor*, *receptor*, and *enzyme*) are even stronger indicators of a protein name. We currently tag words as feature terms if we find them in our list of about 50 such words.

3.2 Lexical analysis of core terms

A core term constitutes the nucleus of a protein name. These terms are the parts of a protein name that show the closest resemblance to regular proper names in that the principles for their coining vary, and often are rather arbitrary. As candidates for these terms we pick words ending in *-ase* and *-in*, or strings with characteristics typical of protein names, i.e., strings containing instances of upper case letters or numbers, found in names of proteins like *HsMad2* and *U3-55k*. Furthermore, as all protein names do not conform to the patterns above, words are dubbed core terms if they are found in a list of established protein names such as *interferon*.

Two general filters are applied to these terms to avoid overgeneration: Words consisting of $\geq$ 50% non-word characters, and measuring units

are discarded as core terms.

3.3 Lexical analysis of specifiers

Yapex also recognizes a third lexical category, the specifier. Specifiers are terms that often occur in the beginning or end of a protein name to, e.g., specify an individual protein. We treat arabic and roman numerals, letters, greek letter names, and combinations of these as specifiers.

3.4 Applying filters and knowledge bases

To remedy the low precision obtained in the previous step, a set of filters is applied to get rid of false hits. Some filters use regular expression patterns of word suffixes to rule out, e.g., names of chemical substances. Other filters use patterns of whole words/expressions to filter out, e.g., personal names and other parts in bibliografical references, chemical formulas, arithmetic expressions, and amino acid sequences. A third group of pattern matching filters remove the core term annotation on words unlikely to function as core terms: Words, ≥ 6 charcters long consisting solely of upper case letters, or consisting of upper case letters and more than one hyphen are discarded.

Short core terms (≤ 3 characters) get special treatment. Only those found in our short-protein-name knowledge base drawn from SWISS-PROT are considered core terms. All the others are tagged as potential core terms to be used later in the protein name identification process. Core terms resembling regular proper names are treated the same way.

3.5 Finding noun phrases

In order to enhance detection of name boundaries, this step takes advantage of the Functional Dependency Grammar (FDG) parser from Conexor Oy (Tapanainen and Järvinen, 1997). For every noun phrase, we identify the head and its preceding lexical modifiers. This constitutes the minimal noun phrase – the noun phrase without any subordinate noun phrases – and is considered a potentional protein name location.

3.6 Identifying protein names

To identify the protein name we start off by adjoining all specifiers to their preceeding core, potential core, or feature term. Then all external or plural feature terms, their adjoined specifiers, and words without a lexical analysis from Yapex is stripped off from the right edge of the noun phrase. From the left edge, words earlier identified as numerals together with measuring units are stripped off. The remaining part of the noun phrase is considered a potential protein name. It is selected as such if it contains a core term, a strong feature term together with at least one other word token, a feature term with an adjoined specifier, or a potential core term together with a feature term somewhere in the unstripped noun phrase.

3.7 Applying a local dynamic dictionary

The relevant terms in the protein names identified in the previous step are stored in a local dictionary as regular expressions. For every document, the dictionary is used in an additional tagging pass over the text to make possible flexible matching of protein names in noun phrases undetected or misinterpreted by the parser.

4 Evaluation

4.1 Reference and test corpora

From the set of answers obtained by posing the following query to MEDLINE[1], 99 abstracts were drawn randomly to form a reference (training) corpus used during development of Yapex:

    protein binding [Mesh term] AND
    interaction AND molecular

with the parameters *abstract, english, human, publication date 1996-2001*. The reference corpus, as well as the test corpus (described below), were annotated by domain experts connected to the Yapex project.

The test corpus consists of 101 MEDLINE abstracts and it is divided into two distinct parts, the first of which — corpus part one — contains 48 abstracts obtained as part of the result when posing the above query to MEDLINE. Part one contains a total of 1213 annotated protein names. The remaining 53 abstracts of the 101 in the test corpus — corpus part two — correspond to a randomly chosen, re-tagged sub-set

| | CORPUS PART ONE | | CORPUS PART TWO | | FULL CORPUS | |
	YAPEX	KEX	YAPEX	KEX	YAPEX	KEX
SLOPPY	$R = 82.8\%$	$R = 83.5\%$	$R = 80.9\%$	$R = 83.4\%$	$R = 82.1\%$	$R = 83.5\%$
	$P = 82.9\%$	$P = 86.0\%$	$P = 85.3\%$	$P = 76.2\%$	$P = 83.8\%$	$P = 82.1\%$
	$F = 82.9\%$	$F = 84.7\%$	$F = 83.0\%$	$F = 79.6\%$	$F = 82.9\%$	$F = 82.8\%$
PNP	$R = 74.0\%$	$R = 68.9\%$	$R = 73.3\%$	$R = 60.1\%$	$R = 73.7\%$	$R = 65.3\%$
	$P = 73.4\%$	$P = 47.7\%$	$P = 77.8\%$	$P = 40.1\%$	$P = 75.1\%$	$P = 44.5\%$
	$F = 73.7\%$	$F = 56.4\%$	$F = 75.5\%$	$F = 48.1\%$	$F = 74.4\%$	$F = 52.9\%$
STRICT	$R = 67.1\%$	$R = 43.2\%$	$R = 65.3\%$	$R = 37.5\%$	$R = 66.4\%$	$R = 41.1\%$
	$P = 67.2\%$	$P = 44.5\%$	$P = 68.8\%$	$P = 34.3\%$	$P = 67.8\%$	$P = 40.4\%$
	$F = 67.1\%$	$F = 43.8\%$	$F = 67.0\%$	$F = 35.8\%$	$F = 67.1\%$	$F = 40.7\%$
LEFT OR RIGHT	$R = 74.7\%$	$R = 57.4\%$	$R = 72.9\%$	$R = 54.3\%$	$R = 74.0\%$	$R = 56.2\%$
	$P = 74.7\%$	$P = 59.1\%$	$P = 76.9\%$	$P = 49.7\%$	$P = 75.5\%$	$P = 55.3\%$
	$F = 74.7\%$	$F = 58.2\%$	$F = 74.9\%$	$F = 51.9\%$	$F = 74.8\%$	$F = 55.8\%$
LEFT	$R = 71.6\%$	$R = 60.9\%$	$R = 71.9\%$	$R = 65.4\%$	$R = 71.7\%$	$R = 62.6\%$
	$P = 71.7\%$	$P = 62.7\%$	$P = 75.8\%$	$P = 59.8\%$	$P = 73.2\%$	$P = 61.5\%$
	$F = 71.7\%$	$F = 61.8\%$	$F = 73.8\%$	$F = 62.5\%$	$F = 72.5\%$	$F = 62.1\%$
RIGHT	$R = 77.7\%$	$R = 53.8\%$	$R = 74.0\%$	$R = 43.3\%$	$R = 76.3\%$	$R = 49.9\%$
	$P = 77.8\%$	$P = 55.4\%$	$P = 78.0\%$	$P = 39.6\%$	$P = 77.9\%$	$P = 49.1\%$
	$F = 77.8\%$	$F = 54.6\%$	$F = 75.9\%$	$F = 41.3\%$	$F = 77.1\%$	$F = 49.5\%$

Table 1: Results for Yapex and KeX given in recall (R), precision (P), and F-score (F).

of the GENIA corpus (Collier et al., 1999) containing 723 annotated protein names. The reference and test corpora are mutually exclusive.

4.2 Notions of correctness

This evaluation presents performance figures for Yapex and KeX[2] on the test corpus using four different notions of correctness. The performance is measured according to the following different notions of correct matching:

SLOPPY: If any token of the proposed hit, as suggested by the tagger, matches some token of the answer key, constructed by domain experts, the hit is counted as a match.

PROTEIN NAME PARTS (PNP): Each token of the hit that matches any token of the answer key is counted as one match. This is a quantification of the SLOPPY match, that gives the degree of overlap between the proposed hit and the answer key.

STRICT: If a proposed hit matches one answer key exactly, the hit is counted as a match.

BOUNDARY:

LEFT: If a proposed hit exactly matches a left boundary in the answer key, the hit is counted as a match.

RIGHT: If a proposed hit exactly matches a right boundary in the answer key, the hit is counted as a match.

LEFT OR RIGHT: If a proposed hit exactly matches any boundary of the answer key, the hit is counted as a match.

4.3 Results

In Table 1, Yapex and KeX are compared in terms of precision, recall and F-score[3]. Looking at the SLOPPY row in the table, we can see that this is the only notion under which Yapex and KeX yield similar figures. The difference between the systems is more obvious, in favour of Yapex, when the other notions of correctness are reviewed — the figures for Yapex are substantially better when measuring the taggers' performance in terms of PNP, STRICT, LEFT, RIGHT and LEFT OR RIGHT. We notice also that it is

---

[2]KeX is a freely available protein name tagger based on the algorithms presented by Fukuda et al. (1998). KeX can be downloaded from http://www.hgc.ims.u-tokyo.ac.jp/service/tooldoc/KeX/intro.html.

[3]F-score is a measure combining precision and recall:

$$F = \frac{(\beta^2 + 1)PR}{(\beta^2 P + R)}$$

where β is a parameter that represents the relative importance of Precision (P) and Recall (R), in our case equally important.

only under the SLOPPY condition that KeX performs close to the results reported in de Bruijn and Martin (2000), but not at all close to what its authors reported in (Fukuda et al., 1998).

Both taggers appear to be stable in the sense that each tagger exhibits similar figures for both precision and recall in any given row in Table 1, with one exception — the difference between recall and precision for KeX under the PNP notion. This, in combination with the results under the SLOPPY condition, suggests that KeX's matches are too long; KeX's high recall and precision under SLOPPY tells us that KeX's suggestions are located close to the correct ones without to many false suggestions entirely outside. Still, KeX gives a lot of false suggestions when it comes to protein name parts.

Visualizing the F-scores for the full corpus evaluation in Figure 1, it is clear that both a STRICT and a PNP definition of a match favours the Yapex system. The result under the PNP condition clearly shows that the overlap between the proposed hits and the corresponding answer keys is remarkably higher for Yapex than for KeX, i.e., Yapex will find more of the protein name parts. We believe that this is due to the ability of the FDG parser to analyse noun phrases well, and thereby predict the boundaries of protein names.

When looking at the result under the STRICT condition, the impression remains the same, suggesting that Yapex is much better at finding the exact edges of the protein names. This is also shown by the result under the LEFT, RIGHT, and LEFT OR RIGHT conditions in Table 1. In fact, this difference is further emphasized if we look at only the correct hits under the SLOPPY condition. Looking at the result this way (Figure 2), we find that Yapex recognizes the correct left boundary in 87.4% of all cases, while the figure for recognising the correct right boundary is a bit higher; 93%. The same figures for KeX is 75% for the left boundary and 59.8% for the right. Thus, in contrast to Yapex, the KeX system appears to correctly recognise the left boundary more often than it does the right boundary. Further, given a SLOPPY hit, Yapex finds one of the left and right boundaries in 90.2% of the cases, while the same figure for KeX is 67.4%. The difference between Yapex and KeX is even greater in the case of the sys-
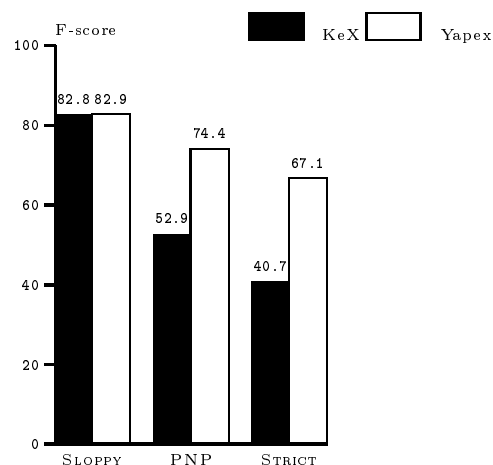

Figure 1: F-score for Yapex and KeX when evaluated on the full corpus along the SLOPPY, PNP and STRICT notions.
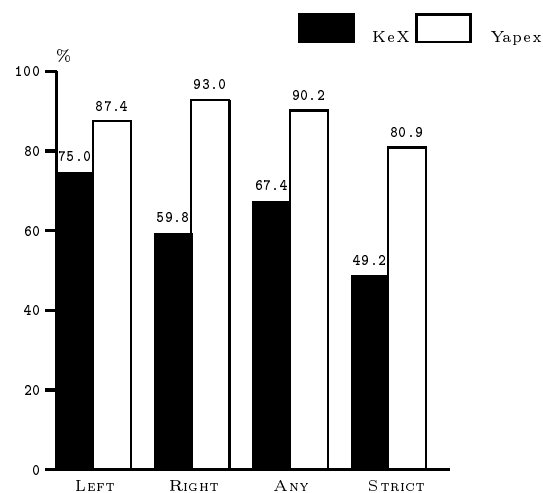

Figure 2: Given a SLOPPY hit, this chart shows the probability of finding protein name boundaries for Yapex and KeX.

tems correctly matching both the left and right boundaries (i.e., STRICT) of a protein name under the SLOPPY condition; 80.9% and 49.2% for Yapex and KeX, respectively.

5 Discussion

To problematize the metrics of recall and precision, we have choosen to evaluate along several notions of correctness. What is relevant to annotate varies with the intended application, and different methods of evaluation can highlight characteristics of competing systems. PNP is a relevant measure for this kind of named ter-

minology where even human domain experts argue about the boundaries of names, since it gives an idea of how much of the multi-word proteins the systems match.

We believe that by equipping Yapex with capabilities of elaborate syntactic analysis, it performs better in recognising protein names with respect to boundaries as well as content, than a system like KeX that does not explicitly exploit syntax. There is nothing surprising about a syntatic parser being able to aid in the detection of protein names; names cannot be found anywhere but in noun phrases. Given a perfect parser that identifies minimal noun phrases, the problem would be reduced to deciding if the noun phrase is a protein name or not. It should be noted though, that we use the FDG parser without modification; it has not been trained to handle this quite specific subdomain of text. Our technique of boosting the identification of protein names by using the Local Dynamic Dictionary finds noun phrases that were not correctly analysed as such by the parser.

What notion of correctness to actually choose to describe the performance of a protein name tagger depends on the setting in which it will be used; in our case, the tagger will be used in a browsing aid, connecting protein names in MEDLINE abstracts with the SWISS-PROT database. Since the query to SWISS-PROT can be made in a way that does not require all parts of the tagged protein name to be present in a SWISS-PROT entry to yield a match, it is not crucial that the tagger achieves perfect matches of the protein names. Thus, in our case, a figure obtained with the SLOPPY notion may suffice to describe the performance of the tagger. In an information extraction setting where the goal is to automatically build a high quality database, it would be more important to find the exact boundaries of the protein names, hence, such an application would benefit from a description along the STRICT or BOUNDARY notions.

It is hard to compare two systems like Yapex and KeX and still maintain a balanced record of result — there is always a risk that the test data is biased towards one of the systems. In this particular case, the domain experts that annotated the test corpus were also involved in discussing the development of Yapex, thus the annotators' definition of what constitutes a protein name

is likely to favour Yapex over KeX. It is possible, e.g., that KeX's low performance under the STRICT, and especially the RIGHT condition is due to a target definition that includes parts of proteins, such as protein *sites* and *domains*. Solving problems like this calls for researchers performing similar studys in the field to clearly state their definitions of what is considered relevant for solving a particular task. Ideally, the research community should strive for shared and open resources. The GENIA project (Collier et al., 1999) is an effort in this direction, but unfortunately, the subclasses of the GENIA protein ontology turned out to be incompatible with our definition of protein names.

6   Conclusions

The LEFT OR RIGHT, LEFT, RIGHT and STRICT notions are suitable for describing a system's behaviour in a setting such as information extraction, where it is crucial that the terms searched for are exactly matched in their entirety. The SLOPPY notion describes a system's ability to find at least some part of the target, and it is sufficient only when the output from the system at hand is to be used in a setting where perfect matches are not crucial.

A combination of the SLOPPY notion and the BOUNDARY one (as in Figure 2) is good for illustrating how well a system is able to delimit a match once it has got a hold of one of the parts of the term searched for.

Presenting results using PNP is suitable for highlighting the system's ability to cover multi-word names.

By using these new notions of correctness — PNP, STRICT and the variants of BOUNDARY — in addition to the commonly used SLOPPY notion, we have illustrated that it is possible to shed light on different aspects of the performance of protein name taggers. Taking into consideration the nature of protein names as such, i.e., the way they are constructed and behave, lead us to believe that the notions are suitable also for other kinds of named terminology.

Acknowledgements

References

Amos Bairoch and Rolf Apweiler. 2000. The swiss-prot protein sequence database and its supplement trembl in 2000. *Nucl. Acids. Res.*, 28:45–48.

Christian Blaschke, Miguel A. Andrade, Christos Ouzounis, and Alfonso Valencia. 1999. Automatic extraction of biological information from scientific text: protein—protein interactions. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB'99)*, pages 60–67, Heidelberg, Germany, August 6-10.

Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. 1998. Nyu: Description of the mene named entity system as used in muc-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, Fairfax, VA, USA, April 29 - May 1.

Nigel Collier, Hyun Seok Park, Norihiro Ogata, Yuka Tateishi, Chikashi Nobata, Tomoko Ohta, Tateshi Sekimizu, Hisao Imai, Katsutoshi Ibushi, and Jun-ichi Tsujii. 1999. The genia project: corpus-based knowledge acquisition and information extraction from genome research papers. In *Proceedings of the European Association for Computational Linguistics (EACL) conference.*

Nigel Collier, Chikashi Nobata, and Jun-ichi Tsujii. 2000. Extracting the names of genes and gene products with a hidden markov model. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, pages 201–207, August.

Berry de Bruijn and Joel Martin. 2000. Protein name tagging. Presented as a poster at the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB'00).

Ken-ichiro Fukuda, Tatsuhiko Tsunoda, Ayuchi Tamura, and Toshihisa Takagi. 1998. Toward information extraction: Identifying protein names from biological papers. In *Proceedings of the Pacific Symposium on Biocomputing (PSB'98)*, pages 705–716, Maui, Hawaii, January 4-9.

Kevin Humphreys, George Demetriou, and Robert Gaizauskas. 2000. Two applications of information extraction to biological science journal articles: Enzyme interactions and protein structures. In *Proceedings of the 5th Pacific Symposium of Biocomputing*, pages 72–80.

Chikashi Nobata, Nigel Collier, and Jun-ichi Tsujii. 1999. Automatic term identification and classification in biology texts. In *Proceedings of the Natural Language Pacific Rim Symposium (NLPRS'2000)*, pages 369–374, November.

Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71, Washington D.C., April. Association for Computational Linguistics.

James Thomas, David Milward, Chirtos Ouzounis, Stephen Pulman, and Mark Carroll. 2000. Automatic extraction of protein interactions from scientific abstracts. In *Proceedings of the Pacific Symposium on Biocomputing (PSB 2000)*, pages 538–549, Oahu, Hawaii, January 4-9.